Department of Computer Science

Advances in Weakly Supervised Learning of Morphology

Oskar Kohonen

Aalto University

DOCTORAL DISSERTATIONS

Aalto University publication series
DOCTORAL DISSERTATIONS 91/2015

Advances in Weakly Supervised Learning of Morphology

Oskar Kohonen

A doctoral dissertation completed for the degree of Doctor of Science in Technology to be defended, with the permission of the Aalto University School of Science, at a public examination held at the lecture hall T2 of the school on 26 August 2015 at 12.

Aalto University
School of Science
Department of Computer Science
Computational Cognitive Systems group

Supervising professor
Distinguished Prof. Emeritus Erkki Oja

Thesis advisor
D.Sc.(Tech) Krista Lagus

Preliminary examiners
Prof. Chris Dyer
Dr. Filip Ginter

Opponents
Prof. Lars Borin
University of Gothenburg, Sweden

Aalto University publication series
DOCTORAL DISSERTATIONS 91/2015

ISBN 978-952-60-6270-9 (printed)
ISBN 978-952-60-6271-6 (pdf)
ISSN-L 1799-4934
ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)
http://urn.fi/URN:ISBN:978-952-60-6271-6

Unigrafia Oy
Helsinki 2015

Finland

441 697
Printed matter

Aalto University

Abstract

Aalto University, P.O. Box 11000, FI-00076 Aalto www.aalto.fi

Author
Oskar Kohonen

Name of the doctoral dissertation
Advances in Weakly Supervised Learning of Morphology

Publisher School of Science

Unit Department of Computer Science

Series Aalto University publication series DOCTORAL DISSERTATIONS 91/2015

Field of research Language Technology

Manuscript submitted 19 January 2014 **Date of the defence** 26 August 2015

Permission to publish granted (date) 30 March 2015 **Language** English

☐ **Monograph** ☒ **Article dissertation (summary + original articles)**

Abstract

Morphological analysis provides a decomposition of words into smaller constituents. It is an important problem in natural language processing (NLP), particularly for morphologically rich languages whose large vocabularies make statistical modeling difficult. Morphological analysis has traditionally been approached with rule-based methods that yield accurate results, but are expensive to produce. More recently, unsupervised machine learning methods have been shown to perform sufficiently well to benefit applications such as speech recognition and machine translation. Unsupervised methods, however, do not typically model allomorphy, that is, non-concatenative structure, for example pretty/prettier. Moreover, the accuracy of unsupervised methods remains far behind rule-based methods with the best unsupervised methods yielding between 50-66% F-score in Morpho Challenge 2010.

We examine these problems with two approaches that have not previously attracted much attention in the field. First, we propose a novel extension to the popular unsupervised morphological segmentation method Morfessor Baseline to model allomorphy via the use of string transformations. Second, we examine the effect of weak supervision on accuracy by training on a small annotated data set in addition to a large unannotated data set. We propose two novel semi-supervised morphological segmentation methods, namely a semi-supervised extension of Morfessor Baseline and morphological segmentation with conditional random fields (CRF). The methods are evaluated on several languages with different morphological characteristics, including English, Estonian, Finnish, German and Turkish. The proposed methods are compared empirically to recently proposed weakly supervised methods.

For the non-concatenative extension, we find that, while the string transformations identified by the model have high precision, their recall is low. In the overall evaluation the non-concatenative extension improves accuracy on English, but not on other languages. For the weak supervision we find that the semi-supervised extension of Morfessor Baseline improves the accuracy of segmentation markedly over the unsupervised baseline. We find, however, that the discriminatively trained CRFs perform even better. In the empirical comparison, the CRF approach outperforms all other approaches on all included languages. Error analysis reveals that the CRF excels especially on affix accuracy.

Keywords morphology, allomorphy, machine learning, unsupervised learning, semi-supervised learning

ISBN (printed) 978-952-60-6270-9 **ISBN (pdf)** 978-952-60-6271-6

ISSN-L 1799-4934 **ISSN (printed)** 1799-4934 **ISSN (pdf)** 1799-4942

Location of publisher Helsinki **Location of printing** Helsinki **Year** 2015

Pages 240 **urn** http://urn.fi/URN:ISBN:978-952-60-6271-6

Aalto-universitetet

Sammandrag

Aalto universitetet, PB 11000, 00076 Aalto www.aalto.fi

Författare
Oskar Kohonen

Doktorsavhandlingens titel
Framsteg inom svagt övervakad inlärning av morfologi

Utgivare Högskolan för teknikvetenskaper

Enhet Institutionen för datateknik

Seriens namn Aalto University publication series DOCTORAL DISSERTATIONS 91/2015

Forskningsområde Språkteknologi

Inlämningsdatum för manuskript 19.1.2014 **Datum för disputation** 26.8.2015

Beviljande av publicerings tillstånd (datum) 30.3.2015 **Språk** Engelska

☐ **Monografi** ☒ **Sammanläggningsavhandling (sammandrag plus separata artiklar)**

Sammandrag

Morfologisk analys delar upp ord i mindre, meningsfulla beståndsdelar. Det är ett viktigt problem inom språkteknologi, särskilt då man behandlar morfologiskt rika språk vars stora vokabulärer försvårar statistisk analys. Det traditionella sättet att framställa morfologiska analysatorer tillämpar regelbaserade metoder. Sådana analysatorer ger noggranna resultat, men är kostsamma att producera. På senare tid har det påvisats att oövervakade maskininlärningsmetoder kan ge tillräckligt noggranna resultat för att vara till nytta i språkteknologiska tillämpningar, t.ex. taligenkänning och maskinöversättning. Oövervakade metoder brukar emellertid inte beakta allomorfi, d.v.s. icke-konkatenativ morfologisk struktur som t.ex. pretty/prettier. Vidare, är oövervakade metoders resultat betydligt mindre noggranna än regelbaserade metoders. De bästa oövervakade metoderna uppnådde 50-66% F-mått i Morpho Challenge 2010.

Vi undersökte dessa problem från två synvinklar som tidigare inte fått mycket uppmärksamhet. För det första föreslog vi en ny extension till den populära oövervakade morfologiska segmenteringsmetoden Morfessor Baseline, för att modellera allomorfi m.h.a. strängtransformationer. För det andra undersökte vi hur svag övervakning påverkar noggrannheten genom att träna maskininlärningsmetoder med en liten annoterad datamängd förutom den stora, icke-annoterade datamängden. Vi föreslog två nya semiövervakade morfologiska segmenteringsmetoder. En semiövervakad extension till Morfessor Baseline, och morfologisk segmentering med conditional random fields (CRF). Vi evaluerar dessa metoder på olika språk med olika morfologiska egenskaper, närmare bestämt på engelska, estniska, finska, tyska och turkiska. De föreslagna metoderna jämförs empiriskt med nyligen framställda svagt övervakade metoder.

För den non-konkatenativa extensionen fann vi att trots att strängtransformationerna som modellen hittade hade hög precision så var deras recall låg. I helhetsevaluationen förbättrar den non-konkatenative extensionen noggrannheten för engelska, men inte för de andra språken. Angående den svaga övervakningen fann vi att den semiövervakade extensionen av Morfessor Baseline förbättrade noggranheten betydligt jämfört med den oövervakande motsvarigheten. Vi fann emellertid också att den diskriminativt tränade CRF-modellen gav ännu bättre noggrannhet. I den empiriska jämförelsen fann vi att för alla inkluderade språk fungerade CRF bäst av de jämförda metoderna. Felanalys visade att CRF var särskilt noggrann då det gäller affix.

Nyckelord morfologi, allomorfi, maskininlärning, oövervakad inlärning, semiövervakad inlärning

ISBN (tryckt) 978-952-60-6270-9	**ISBN (pdf)** 978-952-60-6271-6	
ISSN-L 1799-4934	**ISSN (tryckt)** 1799-4934	**ISSN (pdf)** 1799-4942
Utgivningsort Helsingfors	**Tryckort** Helsingfors	**År** 2015
Sidantal 240	**urn** http://urn.fi/URN:ISBN:978-952-60-6271-6	

Preface

This work has been performed over a number of years, and always at the same department. For the longest part of the process it was called the Adaptive Informatics Research Centre at the Information and Computer Science department. Recently, the department was merged into the department of Computer Science. On a day to day basis the work was performed in the research group on computational cognitive systems lead by Prof. Timo Honkela. The work has been funded by the Academy of Finland through the funding of the Academy research fellowship of Krista Lagus, the Helsinki Doctoral Programme in Computer Science (Hecse), the Academy of Finland research project on Multimodal Language Technology, and the department of Information and Computer Science. Additional support was provided by the Graduate School of Language Technology in Finland (Langnet) as well as the Nokia Foundation.

I would like to thank my supervisor Distinguished Prof. Emeritus Erkki Oja who provided excellent guidance, without which this dissertation would most likely still not be completed. I would also like to thank my advisor Dr. Krista Lagus for investing time in advising me, particularly in the early stages of this work. I am also grateful that I had the opportunity to work with Dr. Harri Valpola whose project taught me enormously about machine learning. I would also like to thank Prof. Timo Honkela for originally getting me involved with natural language processing, as well as many valuable discussions on diverse topics over the years.

The pre-examiners of this dissertation, Prof. Chris Dyer and Dr. Filip Ginter, provided thoughtful comments and valuable suggestions.

I was fortunate to receive comments on the manuscripts from several people: Dr. Mathias Creutz, Dr. Sami Virpioja, Teemu Ruokolainen, Dr. Tiina Lindh-Knuutila, and Mathias Berglund all contributed valuable viewpoints.

I would also like to thank my research collaborators over the years. Especially my closest collaborators, Dr. Sami Virpioja and Teemu Ruokolainen with whom I have spent many good hours thinking about and working on how to slay some particular morphological dragon. In addition to everyone mentioned above, I'm grateful for the opportunity to work with my other co-authors: Mikaela Klami, Laura Leppänen, Prof. Mikko Kurimo, Dr. Kairit Sirts, and Stig-Arne Grönroos.

Dr. Mark van Heeswijk generously shared his recent experience of the process of finalizing the dissertation.

In addition to the ones already mentioned, I would also like to thank my friends and collegues in the computational cognitive systems research group: Jaakko Väyrynen, Matti Pöllä, Mari-Sanna Paukkeri, Paul Wagner, Ilari Nieminen, Marcus Dobrinkat, Eric Malmi, Juha Raitio, Srikrishna Raamadhurai, and Tommi Vatanen. In addition, I would like to thank everyone at the department with whom I have had the pleasure to work and spend time. Thank you!

Finally, I would like to thank my family for support. First, and foremost my wonderful wife Ina Marie for having patience with this project as well as being a genius at planning our work weeks to make space for work and keep our children happy. Moreover, I would like to thank our parents: Eivor, Esa, Christel, and Per Erik for many times of looking after our children while I was writing, writing, writing. Thank you, you are the best!

Espoo, June 16, 2015,

Oskar Kohonen

Contents

List of Publications

This thesis consists of an overview and of the following publications which are referred to in the text by their Roman numerals.

I Oskar Kohonen, Sami Virpioja, and Mikaela Klami. Allomorfessor: Towards Unsupervised Morpheme Analysis. In *Evaluating Systems for Multilingual and Multimodal Information Access: 9th Workshop of the Cross-Language Evaluation Forum, CLEF 2008, Revised Selected Papers, volume 5706 of Lecture Notes in Computer Science*, Aarhus, Denmark, pages 975-982, September 2009.

II Sami Virpioja, Oskar Kohonen, and Krista Lagus. Unsupervised Morpheme Analysis with Allomorfessor. In *Multilingual Information Access Evaluation I. Text Retrieval Experiments, CLEF 2009, volume 6241 of Lecture Notes in Computer Science*, Corfu, Greece, pages 609-616, September 2010.

III Sami Virpioja, Oskar Kohonen, and Krista Lagus. Evaluating the Effect of Word Frequencies in a Probabilistic Generative Model of Morphology. In *Proceedings of the 18th Nordic Conference of Computational Linguistics NODALIDA 2011*, Riga, Latvia, pages 230-237, May 2011.

IV Oskar Kohonen, Sami Virpioja, and Krista Lagus. Semi-Supervised Learning of Concatenative Morphology. In *Proceedings of the 11th Meeting of the ACL Special Interest Group on Computational Morphology and Phonology*, Uppsala, Sweden, pages 78-86, July 2010.

V Teemu Ruokolainen, Oskar Kohonen, Sami Virpioja, and Mikko Kurimo. Supervised Morphological Segmentation in a Low-Resource Learning Setting using Conditional Random Fields. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning (CoNLL)*, Sofia, Bulgaria, pages 29-37, August 2013.

VI Teemu Ruokolainen, Oskar Kohonen, Sami Virpioja, and Mikko Kurimo. Painless Semi-Supervised Morphological Segmentation using Conditional Random Fields. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, Gothenburg, Sweden, pages 84-89, May 2014.

VII Teemu Ruokolainen, Oskar Kohonen, Kairit Sirts, Stig-Arne Grönroos, Sami Virpioja, and Mikko Kurimo. A Comparative Study on Semi-Supervised Morphological Segmentation. *Submitted*, Computational Linguistics, 27 pages, 2014.

Author's Contribution

Publication I: "Allomorfessor: Towards Unsupervised Morpheme Analysis"

The present author, jointly with Sami Virpioja, developed the Allomorfessor model, implemented it, and performed experiments. Regarding method development, the present author, particularly, developed the string transformation class and the search algorithms for transformed words. The publication was written jointly by all authors.

Publication II: "Unsupervised Morpheme Analysis with Allomorfessor"

The present author, jointly with Sami Virpioja, further developed the Allomorfessor model, performed the additional experiments in this paper. The publication was written jointly by all authors.

Publication III: "Evaluating the Effect of Word Frequencies in a Probabilistic Generative Model of Morphology"

The extension proposed in the paper was developed, implemented, experimented with jointly by the present author and Sami Virpioja. The present author also participated in the writing of the publication.

Publication IV: "Semi-Supervised Learning of Concatenative Morphology"

The extension proposed in the paper was developed, implemented, experimented with jointly by the present author and Sami Virpioja. The present author also participated in the writing of the publication.

Publication V: "Supervised Morphological Segmentation in a Low-Resource Learning Setting using Conditional Random Fields"

The present author was responsible for the experimental setup employed in the work, the Morfessor-related experimentation and reporting in the paper. The present author, in addition, participated in the writing of the publication.

Publication VI: "Painless Semi-Supervised Morphological Segmentation using Conditional Random Fields"

The present author set up the experimental setting, implemented the baseline methods, and participated in the writing of the publication.

Publication VII: "A Comparative Study on Semi-Supervised Morphological Segmentation"

The present author implemented the experimental setup and performed the experimental runs with the Morfessor Baseline methods. Moreover, the present author developed and performed the novel error analysis presented in the paper. The paper was mostly written jointly by Teemu Ruokolainen and the present author, with the former having a somewhat larger contribution. The present author particularly contributed to the systematization of the literature presented in the paper.

1. Introduction

Morphology is the subfield of linguistics that studies how words are formed. In abstract terms, morphology can be defined as the study of the systematic covariance between the forms of words and their meanings [Haspelmath, 2002]. For example, the variation in form for the words *car*/*cars* and *door*/*doors* expresses a corresponding variation in meaning; namely, between singular and plural number. Morphological analysis utilizes such systematicity by taking as input the form of a word and producing as output a reading consisting of smaller constituent units that are related to the meaning of the word.

Meanwhile, the field of natural language processing is concerned with building computational systems that process natural language automatically. Examples of such systems include information retrieval, to find documents based on search queries, speech recognition, to transcribe spoken language into written language, and machine translation, to translate a sentence from one language into another. In recent years, such systems have been constructed increasingly with statistical methods [Manning and Schütze, 1999, Manning et al., 2008]. In an abstract sense, statistical methods reformulate these problems into one of estimating a probabilistic mapping between two representations: for information retrieval between queries and documents; for speech recognition between spoken and written sentences, and; for machine translation between sentences in different languages.

Typically, such statistical models utilize words as their basic units. For many languages, including English, word-based models perform well. In contrast, for languages with a rich morphology, word-based model suffer from problems with data sparsity. This is because a rich morphology produces so many different word-forms from a single root. For example, a single Finnish verb can produce over 20,000 inflected forms [Arppe, 2005].

Consequently, in morphologically rich languages, many of the words that are encountered when applying the model were seen rarely, if at all, in the data set used to estimate the model [Kneissler and Klakow, 2001, Kurimo et al., 2006b].

As an alternative to word-based models, it is possible to utilize morphological units. A morphological concept well suited for this purpose is that of **morphemes**, defined as the smallest meaning bearing units in language [Hockett, 1954, Matthews, 1991]. In this view, words are constructed out of one or more morphemes. For example, the word *unlimited* can be segmented as $un \circ limit \circ ed$.

To utilize morphological units, automatic morphological analysis is required. Rule-based systems for morphological analysis have existed for a long time, and provide accurate results [Koskenniemi, 1983, Kaplan and Kay, 1994, Karttunen and Beesley, 2005]. Developing the required rule-sets, however, is labor intensive, and consequently many languages lack freely available, rule-based morphological analyzers.

An alternative approach that has become popular recently is to apply unsupervised machine learning to learn morphology from a large list of unannotated words in the target language (see e.g. Hammarström and Borin [2011], Roark and Sproat [2007], Creutz and Lagus [2007], Goldsmith [2001]). The benefit of unsupervised methods is that they are very easy and inexpensive to apply to any language. The drawback is that their accuracy is far behind that of rule-based methods. Despite lower accuracy, the output of unsupervised methods has been found empirically useful in a wide range of applications, including speech recognition [Hirsimäki et al., 2006, Narasimhan et al., 2014], information retrieval [Turunen and Kurimo, 2011], machine translation [de Gispert et al., 2009, Green and DeNero, 2012], and word representation learning [Luong et al., 2013, Qiu et al., 2014, Botha and Blunsom, 2014].

1.1 Machine Learning of Morphology – Allomorphy and Weak Supervision

This work takes as a starting point the previous work on unsupervised learning of morphology. Our goal is to improve the quality of morphological analysis while preserving the inexpensive nature and ease in application of the original unsupervised methods. To serve this goal we will address two areas which have not received much attention in the past.

First, we consider non-concatenative structure, that is, **allomorphy** Most unsupervised methods are limited to producing a morphological segmentation, that is, they segment words into morphological units. Meanwhile, language contains non-concatenative morphological structure that cannot be modeled well by a segmentation alone. For example, for *white–whiter*, there is no single segmentation that allows expressing that both the segments *white* and *er* are present. Moreover, for *prettier* it is also impossible to express the presence of a relation to *pretty* with a segmentation, because of the letter-change. In morphology, such structures are addressed by distinguishing between morphemes as abstract units and their surface forms **morphs**. For example, *pretty* and *pretti* are said to be **allomorphs** of the same abstract morpheme.

We approach this problem by proposing a novel extension to the Morfessor Baseline-method [Creutz and Lagus, 2002, 2007], from segmentation only, to also associating segments through string transformations. This approach models latent morphemes that are then expressed as different allomorphs.

Second, we address the accuracy of the previously proposed methods. There are several potential approaches to improve performance. One approach is simply further model development. We will consider an alternative approach, namely annotating a small data set. Both further model development and annotation are time-consuming. However, it is currently not well known which approach is more cost-effective. We will address this question by studying semi-supervised learning with a small amount of annotated data. We refer to this setting as weakly supervised learning. We require the training sets to be small in order to keep the methods easily applicable to new languages.

1.2 Contributions of this thesis

This thesis has the following contributions.

First, we develop a novel method for unsupervised learning of morphological analysis in the presence of stem-allomorphy. The method extends the generative probabilistic model of Morfessor Baseline [Creutz and Lagus, 2002, Creutz et al., 2007], and enables the learning of non-concatenative variation in stems by adding string transformations to the generative model. This model is detailed in Publication I and Publication II. The method was applied and evaluated empirically in the Morpho

Challenge competitions 2008 and 2009 on five different languages [Kurimo et al., 2009a,b].

Second, we develop novel methods for weakly supervised learning of morphological segmentation, in particular for a semi-supervised learning setting with a small number of annotated words and a vast amount of unannotated words. In Publication III we develop a hyperparameter formulation for the unsupervised Morfessor Baseline method [Creutz and Lagus, 2002, Creutz et al., 2007] that allows employing labeled data to control how much the method segments on average. This formulation is expressed through a weight in the objective function. In Publication IV we develop a novel extension of the generative probabilistic model of Morfessor Baseline [Creutz and Lagus, 2002, Creutz et al., 2007] to semi-supervised learning by further extending the weighted objective function. The novel method outperforms its unsupervised baseline by a wide margin. In Publication V and Publication VI we present a novel application of structured classification methodology to the problem. In particular, we apply conditional random fields (CRF) [Lafferty et al., 2001], first in a supervised fashion in Publication V, and then in Publication VI we extend to semi-supervised learning. The semi-supervised extension is based on augmenting the feature set of the supervised classifier with the output of unsupervised morphological segmentation methods. We find that surprisingly small annotated sets are sufficient for the semi-supervised structured classifier to yield superior performance over the generative models proposed in Publication IV.

Finally, Publication VII provides a systematization of the current state of art approaches to weakly supervised morphological segmentation. Publication VII includes a detailed empirical comparison on four languages between the methods of Publication IV, Publication V, Publication VI, as well as other recently proposed methods, including two methods from the Adaptor Grammar framework [Sirts and Goldwater, 2013] and Morfessor FlatCat [Grönroos et al., 2014]. We find that, for all languages and data sizes at the minimum of 100 annotated words, the best performance is achieved with the structured classification approach proposed in Publication VI.

1.3 Thesis Outline

The remaining part of this dissertation is structured as follows:

Chapter 2, Morphological Analysis In this background chapter, we discuss the linguistic theory of morphology and review computational tasks related to morphological analysis.

Chapter 3, Machine Learning Preliminaries We then discuss machine learning methodology to the extent that will be required in later chapters.

Chapter 4, Related Segmentation Methods This chapter describes the segmentation methods that are either extended or applied in this work. We review the unsupervised morphological segmentation method Morfessor, in particular, the variant that Creutz et al. [2007] refer to as Morfessor Baseline. This Morfessor variant is extended in the later chapters to the learning of allomorphy and to semi-supervised learning. We also review segmentation with the sequence labeling method conditional random fields [Lafferty et al., 2001].

Chapter 5, Unsupervised Learning of Allomorphy In this chapter we review the problem of learning allomorphy and our proposed extension of Morfessor Baseline [Creutz and Lagus, 2002, Creutz et al., 2007] to allow allomorphic variation through string transformations, as introduced in publications Publication I and Publication II. We also review empirical results and discuss implications of the work.

Chapter 6, Semi-Supervised Learning of Morphological Segmentation This chapter discusses the weakly supervised learning of morphological segmentation in a semi-supervised setting with a small annotated data set and a large set of unannotated words. We begin, following Publication III and Publication IV, to extend Morfessor Baseline [Creutz and Lagus, 2002, Creutz et al., 2007] to semi-supervised learning. We then apply conditional random fields (CRFs) to morphological segmentation, following Publication V and Publication VI. Finally, following Publication VII, we review an empirical comparison of the methods introduced in this thesis, as well as comparison to other recently proposed methods [Sirts and Goldwater, 2013, Grönroos et al., 2014].

Chapter 7, Conclusions In this chapter we draw conclusions from the results and review their implications.

2. Morphological Analysis

In this chapter we discuss morphology as a linguistic phenomenon and the corresponding computational problems related to morphological analysis. We start by discussing morphology, including its concepts and terminology, in Section 2.1. We then turn to describe automatic morphological analysis tasks in natural language processing as well as the evaluation of such systems in Section 2.2.

2.1 Morphology

Morphology is concerned with the grammatical structure of words. It is situated between syntax that focuses on the structure of sentences, and phonology which is related to the sound-structure of language. While these are traditionally separate areas of linguistic study, they do often interact in practice. We will try to cover the interactions to the extent it is necessary for the discussion, nevertheless, focusing on the morphological aspects. Morphology can be studied both for spoken and written language. The spoken form can be considered primary, since learning spoken language happens naturally while learning written language requires explicit teaching which can only take place after spoken language is already established. Nevertheless, because we are primarily focused on automatic processing of texts, that is written language, we will focus the presentation below on written language. However, it is usually possible to substitute the written form, that is the **orthography**, for corresponding spoken form, the **phonology**.

Morphology can be more precisely characterized following the concise definition of Haspelmath and Sims [2010]:

> Morphology is the study of systematic covariation in the form and meaning of

words

As an example of covariation, consider that the words *car* and *cars* are similar both in the form, sharing the substring *car*, and their meaning as both invoke the same class of entities in the world. In contrast, the words *car* and *care* are similar in their form but not in their meaning and are, therefore, not morphologically related.

Morphology is a diverse field of study and we will here briefly describe some aspects relevant to the machine learning problems in the later chapters. We begin by describing the basic terminology, then review different morphological models, then how morphological systems vary across languages, and finally we discuss motivations for the definitions as well as some alternative ideas. For a more detailed treatment, see [Hockett, 1954, Matthews, 1991, Karlsson, 2002, Manning and Schütze, 1999, Roark and Sproat, 2007, Haspelmath and Sims, 2010].

Morphology - Basic Terms

As in many other linguistic fields, morphology employs the distinction between **surface forms** that we may encounter in a text and **abstract** units that are part of the language system but require language-specific knowledge to observe. We will write abstract units in upper case in contrast to concrete units, that is surface forms, that are written in lowercase.

Before examining smaller constituents of words, we need to define some basic terminology on the word level. First, we define some terms related to observing words in a text. Word **tokens** are words that we encounter in a text, whereas word **types** are unique word strings. For example, the sentence *"The car is in the parking lot"*, contains two tokens of the word type *the*. If the language marks word boundaries, as is the case with the space character used in most Western languages, word tokens and word types are easily identified programmatically on the computer. Second, we discuss some word terminology that is not simply calculated from the surface form but, rather, requires detailed knowledge of the language. Words can be grouped by **lexeme**, that is, the same word in the abstract or dictionary sense. The lexeme then occurs in different **word-forms**. For example *lives, living*, and *lived* are word forms of the lexeme *LIVE*. Typically, one represents the lexeme by choosing a particular word-form, for example *live*. This representative form is referred to as the **lemma**, **basic form**, or **base form**.

Because morphology and syntax interact to some degree, we must also discuss the syntactic notion of **part of speech** which concerns the function of a word in a sentence. Examples of parts of speech categories include nouns, verbs and adjectives. Parts of speech are context-dependent and the same word type can take on different parts-of-speech in different contexts. The reason for this is that word-forms may be **homonymous**. Different word-forms of the same lexeme or word-forms of two different lexemes look identical. Consequently, there are typically fewer word types than the number of lexemes and word-forms would imply. For example, the English word type *trying* may be either a word-form of the adjective *TRYING*, such as in *"This was a trying walk"*, or a word-form of the verb *TRY*, as in *"He is trying to learn the banjo"*. When there is homonymy between different forms of the same lexeme it is referred to as **syncretism**. A homonymous word type implies that there are several different paths through which it can be produced. In other words, its analysis is **ambiguous**.

Word-forms are typically constructed by adding material to another word-form. We can classify parts of words based on whether the part is free to occur by itself, in which case it is called a **stem**, or must always be bound to some other part of the word, referred to as an **affix**. A stem can itself have further morphological structure, consisting of further stems and affixes, or it may be minimal, in which case it is called a **root**. Affixes attaching at the end of a stem are called **suffixes**; at the beginning of the stem **prefixes**; in the middle of the stem **infixes**; and at both ends of the stem **circumfixes**. Stems and affixes do not combine arbitrarily, but exhibit **selectional preferences**, such that only some kinds of stems combine with particular affixes. For example, the English verb stems combine with verb affixes, but not noun affixes. Although the division into stems and affixes is applicable to many languages, there are also different morphological systems. For example, Semitic languages encode lexemes with consonant patterns and the different word-forms of the lexeme are identified by the vowels between the consonants.

Morphological Segmentation

As a first level of morphological analysis we can perform **morphological segmentation** and segment words into their stems and affixes. For example *car* ∘ *s, live* ∘ *s, liv* ∘ *ing*. From the previous examples we can see that the two first forms separate nicely into stem and affix, but for

liv ∘ *ing*, some material is lost from the lexeme *LIVE*. When a word can be formed from stems and affixes without changes, the structure is called **concatenative** or **agglutinative**. In contrast, when the stems and affixes are not combined by simple concatenation the structure is called **non-concatenative** or **fusional**. Generally, non-concatenative structure includes irregular structures, such as *go*/*went*. Often despite a structure being non-concatenative, there is, nevertheless, a **morphological pattern** that is at least partially regular. For instance *sing*/*sang*/*sung* and *fling*/*flang*/*flung*. A common regularity is the **alternation**, where two different variants of a unit occur in complementary distribution, that is, in a particular word-form you see either one variant or the other. For instance the Finnish plural alternates between being marked by *i* or *j*: *talo* ∘ *ja* (houses), *talo* ∘ *j* ∘ *en* (houses'), *talo* ∘ *i* ∘ *ssa* (in houses), and *talo* ∘ *i* ∘ *tta* (without houses).

Morphological Differences between Languages

The morphological systems vary widely between languages. There are two central dimensions of variation among languages that are of particular interest to our discussion: First, different languages employ varying degrees of morphological richness. Languages that infrequently employ morphological structure are referred to as **isolating**, and languages with a rich morphology as **synthetic**. Another dimension of variation among the morphology of languages is between **agglutinative** languages, where the stems and affixes are combined concatenatively, and **fusional languages** where non-concatenative fusion is common. Languages typically employ a mix of both agglutination and fusion, but vary in how often these are employed. Therefore, the characteristics of a particular language can be characterized on a continuum of these characteristics, referred to as the degree of syntheticity and the degree of fusion [Sapir, 1921, Karlsson, 2002].

Morphological Processes

Words are formed through different processes. **Inflection** forms the different word-forms of the same lexeme. In contrast, **word formation** produces new lexemes from existing ones. There are two different mechanisms for word formation: **derivation** takes one lexeme and produces another, modified lexeme; **compounding**, takes two lexemes and joins them together as a new lexeme. To relate these processes to the previously defined terms, we can note that stems express lexemes, whereas

affixes are utilized for both inflection and derivation.

The inflections forming the word-forms of a single lexeme are called a **paradigm**. Inflections are typically related to the syntactic function of the word in the sentence. For example, in English, when combining a verb in present tense with a pronoun in the third person singular the verb takes on a suffix, for instance in *He eats*. Such structure is called **agreement**. Therefore, the forms in a paradigm are usually organized according to their function in the sentence, encoded by **morphosyntactic features**, for example person and number.

Morphological processes are **productive** to a varying degree. Inflectional processes are typically completely productive such that a given form in a paradigm can be formed for any lexeme belonging to its corresponding class. This extends even to new words. Berko [1958] demonstrated that even young children have the ability to construct different forms of newly invented words. For example, the children were shown pictures of a creature and were told it was a 'wug'. They were then shown a picture of two of the creatures and asked to fill in the sentence "two ____", where the correct answer was 'wugs'. In contrast to the productivity of inflectional processes, word formation processes are productive only to varying degrees. Some word formation processes are applicable to many words, whereas others can be applied only to a small number of and words, or even not applied at all outside archaic forms.

Morphological Models

Morphology is an active area of research, and there are many different morphological models. We cover some classical work and discussion that are central to our methods and data sets. We begin by reviewing the two models identified by Hockett [1954], namely the **item-and-process** and the **item-and-arrangement** models. We will then proceed to discuss some more recent developments.

Item-and-process model The item-and-process model centers around the lexeme and describes the process by which the inflected forms are produced. The item-and-process model then describes the formation of the word-forms by listing processes that operate on a **basic form**. This process may be one of adding **affixes**, however in the item-and-process model the affixes are not considered their own units, merely operations on the basic form, and other, non-concatenative, operations are also possible. The item-and-process model describes also the phonological contexts in which

a particular variant is chosen.

Item-and-arrangement model The item-and-arrangement model centers around the notion of a **morpheme**, defined as the smallest meaning-bearing unit in language. Words are thought to be constructed by combining a sequence of morphemes. The morphemes are abstract units, and their surface realizations are referred to as **morphs**. When the morphological structure is concatenative the morphemes can be found with morphological segmentation. For non-concatenative structure, we need to identify the abstract morphemes. For example, for *car* ∘ *ing* we have two morphemes: the lexeme *CARE* and the present participle suffix morpheme. In such non-concatenative cases the same morpheme *CARE* has more than one surface realization, for example *care* and *car*. These are called **allomorphs**. The non-concatenative allomorphic relations can further be divided into **phonological allomorphs** that are phonologically similar, such as the above example, and **suppletive allomorphs** that are not, for example *go*/*went*. Furthermore, suppletion exists on a continuum, where some instances still maintain some similarity, such as *catch*/*caught*.

Discussion

When there are several distinct morphological models available, a central question is which model to choose in a given situation. The item-and-process model is well suited for teaching the classical languages, such as Latin, to which it was first applied. Agglutinative languages, however, are problematic for such word-based models. For example, describing all the 20,000 verb forms in Finnish [Arppe, 2005] with the item-and-process model would be very tedious. For that task, however, morpheme-based models, such as item-and-arrangement, are well suited.

Meanwhile, morpheme-based models also have a number of problems [Haspelmath and Sims, 2010]. First, fusional characteristics create a large number of allomorphs, despite the morphological patterns being quite regular. An example of such patterns are vowel-changes. Another related problem with fusional languages is that when the affixes of morphosyntactic features are combined, a single affix expresses more than one feature. Therefore, a single morpheme for the morphosyntactic feature cannot be established. Second, morpheme-based theories typically assume that if the morphemes are the smallest meaning bearing units, then the meaning of a combination of morphemes should be **composi-**

tional, that is, predictable from the meanings of the individual morphemes. In practice, not all meanings of word-forms can be predicted from the meanings of their parts. This is especially true for derivations. For example, a *read* ∘ *er* is not just a person who reads, but also an academic title and, in the case of *e-book reader*, a device on which to read. Finally, there is psycholinguistic evidence that when people process regular word-forms, they react faster to frequent ones [Baayen et al., 1997]. This implies that the brain stores not only the irregular forms but also the regular forms that could be constructed using rules.

Combining Storage of Word-Forms and Productivity To address the problems of the item-and-process model and the item-and-arrangement models, alternative models have been proposed. Haspelmath and Sims [2010] discuss a different kind of word-based model of morphology that includes productive patterns. In contrast to the item-and-arrangement model, the combined word and pattern model stores many regular word-forms in the lexicon despite their being possible to construct from rules. This resolves problems with non-compositionality of meaning. To resolve the problems for agglutinative languages, this morphological model stores morphological patterns as well. This enables the construction of new forms when needed.

2.2 Computational Morphology and Natural Language Processing

Having reviewed the linguistic aspects of morphology we now turn to computational tasks in morphology. We will focus, especially, on morphological analysis and morphological segmentation as this dissertation is concerned with machine learning models for these tasks. Finally, we will discuss different methods for evaluating morphological analyses

2.2.1 Tasks in Computational Morphology

A central distinction between different morphological analysis tasks is whether the input of the analyzer is an isolated word type or a word token in sentence context. The former is more commonplace, but has the downside that, due to homonymy, the analysis is often ambiguous. We will review three approaches of this kind: stemming, morphological segmentation, and morphological analysis. The experimental work in later chapter will focus on the two latter tasks. We will also briefly review a

task in sentence context, namely morphological tagging.

We denote the input $\mathbf{x}$, and it is a word string, such as *techniques* or *music*. The word string $\mathbf{x} \in \Sigma^*$, that is, it is an arbitrary-length string of characters from the alphabet Σ of the language under study. We denote the correct analysis $\boldsymbol{y}$.

Stemming and Lemmatization

Stemming is a simplistic form of morphological analysis that strives to normalize words, such that all word-forms of the same lexeme would yield the same stem. For English, the Porter stemmer is well-known and performs rather well [Porter, 1980]. It requires only a small number of rules that identify common suffixes and remove them. It also contains simple rules to handle common alternation patterns in English. For instance, between y/i in words such as *pretty*/*prettier*. In general, building the stemming rules becomes complicated for morphologically rich languages. For example a stemmer for Slovene utilizes a suffix list of over 5000 suffixes [Popovič and Willett, 1992].

Lemmatization strives to also identify the lexeme, but does this by mapping the observed word-form to its representative lemma. Lemmatization generally requires detailed knowledge of the lexicon. For morphologically rich languages lemmatization requires so much morphological knowledge that it is typically performed as a subtask of morphological analysis.

Morphological Segmentation

In morphological segmentation the word $\mathbf{x} \in \Sigma^*$ is segmented such that each segment is a morph, the surface form of a morpheme. The true sequence of morphs $\mathbf{y}$ consists of one or more sequences of morphs, since the segmentation may be ambiguous. Each alternative, correct segmentation $\mathbf{y}_k = \mathbf{y}_{k1}, \ldots, \mathbf{y}_{kn}$ consists of a sequence of morphs, and since they are segmentations their concatenation produces the input word $\mathbf{y}_{k1} \circ \cdots \circ \mathbf{y}_{kn} = \mathbf{x}$. A convenient property of segmentation is that an unsupervised algorithm can return a segmentation and, therefore, unsupervised and supervised algorithms can be evaluated head-to-head using identical evaluation procedures.

Morphological Analysis

In morphological analysis the task is to find the morphological units of the word. The chosen unit depends on the underlying morphological theory that is being employed. Common alternatives are *morphemes* and *mor-*

word	morphological analysis	morphological segmentation
auto (car)	auto+N+Sg+Nom	auto
autossa (in car)	auto+N+Sg+Ine	auto ∘ ssa
autoilta (from cars)	auto+N+Pl+Abl	auto ∘ i ∘ lta
autoilta (car evening)	auto+N+Sg+Nom+# ilta+N+Sg+Nom	auto ∘ ilta
maantie (highway)	maantie+N+Sg+Nom	maantie
	maa+N+Sg+Gen+# tie+N+Sg+Nom	maa ∘ n ∘ tie
maanteiden	maantie+N+Pl+Gen	maanteide ∘ n
(of highways)	maa+N+Sg+Gen+# tie+N+Pl+Gen	maa ∘ n ∘ teide ∘ n
sähköauto	sähköauto+N+Sg+Nom	sähköauto
(electric car)	sähkö+N+Sg+Nom+# auto+N+Sg+Nom	sähkö ∘ auto

Table 2.1. Morphological segmentation versus morphological analysis for exemplar Finnish words. The full analysis consists of word lemma (basic form), part-of-speech, and fine-grained labels. The form *maanteiden* employs non-concatenative structure with the lexeme *TIE* (road) appearing as the allomorphs *tie* and *teide*.

phosyntactic tags. The formalization is similar to that of morphological segmentation, but now the $\mathbf{y}_{ij}$ are not strings, but are instead morphological tags generated from the set $\mathcal{Y}$. The tagset consists of lexeme and affix tags. It can be noted that this formalization is very similar to a multi-label classification problem [Tsoumakas and Katakis, 2007]. Morphological analysis differs from multi-label classification in that the correct morpheme sequence $\mathbf{y}$ is often ambiguous and the correct analysis is a sequence rather than a set. Moreover, an unusual property is that the set of morphological tags $\mathcal{Y}$ is generally *open*, that is new tags are added to it over time as new words are formed. A simplification that is taken in most practical systems is to assume a fixed, but large set of morphemes.

A central difference when applying morphosyntactic tags rather than morphemes is that the morphosyntactic tags will be applied even in the absence of any surface form of a morpheme. For example *car* would only contain one morpheme tag *CAR-N*, with *-N* denoting a noun, but would be assigned the morphosyntactic tags *CAR-N* + *Sg*, for singular number.

In contrast to morphological segmentation, an unsupervised algorithm cannot return morphological analysis in the correct tagset $\mathcal{Y}$, as the learner receives as input only a set of words $\mathbf{x}$. Instead, the learner must produce a set of arbitrarily named tags for each proposed morpheme. Consequently, although evaluating supervised morphological analysis is straightforward, evaluating unsupervised morphological analysis is non-trivial.

A rule-based morphological analyzer returns all correct morphological

analyses of a given word. Often the same word type can be produced from several different lexemes, in which case the analyzer returns all of them. Some examples of morphological analysis produced by a recent rule-based analyzer [Pirinen, 2008] employing morphosyntactic tagging are shown in Table 2.1 and contrasted with corresponding morphological segmentations. It can be seen that both the analysis and the segmentation are occasionally ambiguous. Moreover it can be seen that the analysis is much more detailed, returning the lemma of each stem, its part-of-speech tag and also grammatical categories that correspond to the absence rather than presence of a morph. For example *maa+N+Sg+Gen* for the observed morphs *maa+n* in the word *maantie*. Here *maa* is the lemma, *N* the part-of-speech, *Sg* denotes singular number which is not morphologically marked (plural would be), and finally *Gen* denotes Genitive and corresponds to the morph *+n*.

Morphological Tagging

The task of determining the correct alternative analysis of a word token in context is known as **morphological tagging** [Hajič, 2000]. Morphological tagging typically ignores determining the correct lexeme, and concentrates on merely identifying the correct morphosyntactic tags for each word. The task is often approached with similar statistical methods as part-of-speech tagging [Müller et al., 2013, Silfverberg et al., 2014].

Evidently, morphological tagging is also closely related to syntactic analysis of the sentence, and is therefore employed when performing parsing and developing training data for syntactic analysis [Haverinen et al., 2014, Bohnet et al., 2013].

2.2.2 Evaluation

Assuming that we have some automatic method for producing a morphological representation from input words, we need to consider how the performance of such a system can be evaluated. There are two main options. We can either evaluate our model by comparing it to some analysis that we assume is correct, or we can apply our analysis in some task and evaluate the task-specific results. The former is known as **intrinsic** and the latter **extrinsic** evaluation. The characteristics of the two are somewhat different. Next we review two central intrinsic evaluation measures and then return briefly to extrinsic evaluation.

Boundary F1-score

Morphological segmentations can be evaluated by an intrinsic comparison with reference segmentations using **boundary precision**, **boundary recall**, and **boundary F1-score**. The boundary F1-score, or F1-score for short, equals the harmonic mean of precision, the percentage of correctly assigned boundaries with respect to all assigned boundaries, and recall, the percentage of correctly assigned boundaries with respect to the reference boundaries:

$$\text{Precision} = \frac{C(\text{correct})}{C(\text{proposed})} \tag{2.1}$$

$$\text{Recall} = \frac{C(\text{correct})}{C(\text{reference})} \tag{2.2}$$

The Expressions 2.1 and 2.2 for different words can be combined into a single value utilizing either micro or macro averages, that is either giving each segment, or each word, equal weight in the combined score.

The Morpho Challenge measure

The Morpho Challenge competitions [Kurimo et al., 2009b] employ a measure for intrinsic evaluation of morphological analyses. It is intended for comparison to a morpheme-based gold standard, as employed in the competitions. The measure is designed to evaluate an unsupervised method compared to a morpheme-based gold standard, assuming that the unsupervised method does not know the true label set.

The evaluation measure randomly samples a number of focus words from the test set. The sampled focus words are then compared with reference words such that if two words share a morpheme in the reference, they should also share a morpheme in the analysis proposed by the method being evaluated. Precision measures whether the word types that share morphemes in the proposed analysis have common morphemes also in the gold standard. Recall is calculated analogously by swapping the roles proposed and gold standard analyses. The final score is the F-measure, the harmonic mean of precision and recall.

It has been shown that the scores of the MC metric can be artificially elevated by manipulation of the proposed analyses [Spiegler and Monson, 2010]. Other methods have been proposed for evaluating morphological analysis, and such methods are extensively reviewed by [Virpioja et al., 2011].

Extrinsic Evaluation

If the ultimate purpose of the morphological representation is some particular application then evaluating directly in that application is natural. Interestingly, the optimal morphological representations can, however, be task-specific.

For example, Pirkola [2001] discusses utilizing morphological analysis in information retrieval, and suggests that inflections, derivations and compounding need to be considered separately. It is often the case that the derived or compound meaning is not compositional, and therefore, including them as wholes may be preferable.

In machine translation different language pairs may require a different level of detail from the morphological analysis. For example, German frequently employs compound words which translate into several words in English, and this property can be problematic for translation systems. Consequently, determining the optimal granularity of morphological analysis has been attempted by several authors [Koehn and Knight, 2003, Goldwater and McClosky, 2005, Habash and Sadat, 2006, Dyer, 2009].

The task-specific nature of extrinsic evaluation is problematic for morphological analyzer development as a method may perform well in one application and badly in another. This property, however, also opens up the possibility to construct adaptive methods with parameters that enable quick adaptation to different tasks.

3. Machine Learning Preliminaries

The field of machine learning studies the modeling of patterns and the discovery of generalizable properties from *data*. Typically, this means that we can learn some **model** from a **training data set** and then apply that model to data that we have not seen at training time, **test data**, and get some kind of analysis of it. If the generalization is successful, then the analysis should correspond to the correct analysis, where correctness is task-specific. For example, if the task is morphological analysis, then the correct analysis would correspond to a correct morphological analysis of the words in the test set.

In this section we review machine learning concepts, especially from the perspective of probabilistic modeling. Because of the generality of the contents, we do not refer to the literature for each claim, but the presentation is based on [Manning and Schütze, 1999, Jaynes, 2003, Alpaydin, 2004, Bishop, 2006, Sutton and McCallum, 2006, 2012]. We will begin by discussing different learning setups in Section 3.1, where we will also define weakly supervised learning, the focus of this work. Next, in Section 3.2, we will discuss probabilistic modeling. We will then describe graphical models in Section 3.3. Finally, we discuss model selection in Section 3.4.

3.1 Learning Setups

Learning setups and methods can be classified based on what data is available at training time. We will review these below. Methods can also be classified based on how they operate at test time: *transductive* methods can only be applied to data seen at training time, while *inductive* methods can be applied to any data. All methods we employ are inductive. Although the learning settings are general, we will here focus on the morphology learning tasks, and therefore refer to the data as words and

annotation.

3.1.1 Unsupervised Learning

In unsupervised learning the system receives at training time only an **unannotated** set of data, which we denote $\mathcal{U} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(N)}\} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$, where $\mathbf{x}^{(i)}$ denotes the ith sample in the set. Since there is no target defined in the input data, very different unsupervised learning tasks can be performed on the same input data. Unsupervised learning can therefore be exploratory in nature, seeking interesting properties of a data set, rather than attempting to solve a particular task. It is, however, also possible to employ unsupervised techniques to a particular task, such as morphological segmentation.

3.1.2 Supervised Learning

In supervised learning the system receives pairs of input observations and their corresponding annotation. Therefore, the training data set may be written as $\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{N}$. For example, in the case of morphological segmentation, the data contains word strings, such as $\mathbf{x}^{(i)} = \textit{preheated}$, and the annotation contains their correct segmentations, $\boldsymbol{y}^{(i)} = \textit{pre} \circ \textit{heat} \circ \textit{ed}$. The task is to learn to predict the annotation from the input observations, such that this prediction generalizes to data that the system has not seen at training time.

3.1.3 Semi-Supervised Learning

With semi-supervised learning we refer to a setting where we have both annotated data $\mathcal{D}$ and unannotated data $\mathcal{U}$ [Zhu, 2006, Zhu and Goldberg, 2009]. The tasks are generally similar to supervised prediction tasks, and the unannotated data is employed to improve results over purely supervised training on $\mathcal{D}$ alone.

3.1.4 Weakly Supervised Learning

Weak supervision is a term that is used in different senses by different authors. First, the term is used to describe a situation where the labeled data is somehow insufficient. For example, insufficiently small or lacking some of the required classes (see e.g. Ng and Cardie [2003], Zhang [2004], Paşca [2007]). Second, the term is applied when utilizing some

supervision that is not precisely the annotation for the current task, but something related to it (see e.g. [Klementiev and Roth, 2006]).

In this dissertation, we define weakly supervised learning simply as providing only a small amount of labeled data in a semi-supervised setting. In other words, we follow the first definition above. We consider this supervision weak because small annotated sets cannot provide good coverage of the morphological phenomena in the language. For example, it has been suggested that for a good coverage of an English language model 50,000 words are required [Kurimo et al., 2006a]. If the training data contains only 1,000 words, then by necessity many of the stems will be unseen in the data.

A term related to weakly supervised learning employed by several authors is minimally-supervised learning [Yarowsky and Wicentowski, 2000, Wicentowski and Yarowsky, 2003, Riesa and Yarowsky, 2006, Monson et al., 2007, Sirts and Goldwater, 2013] and Publication VII. Unfortunately, that term is also used in several different senses. Generally, the term refers to providing supervision that is easy and inexpensive to collect, similar to the second definition of weak supervision above [Yarowsky and Wicentowski, 2000, Wicentowski and Yarowsky, 2003]. The term has, however, also been used merely for hyperparameter adjustment [Monson et al., 2007]. It has also been utilized to describe what is referred to as the weakly supervised setting in this dissertation. In particular, that usage is employed by Sirts and Goldwater [2013] and in Publication VII.

In this dissertation, we will favor the term weakly supervised, since the term minimally-supervised implies the supervision is minimal in some well-defined sense, whereas our training sets are merely small.

3.2 Probabilistic Modeling

Probabilistic modeling provides a mathematical framework within which machine learning methods can be developed and analyzed. Compared to earlier machine learning methods based on learning rules and objective functions, the probabilistic framework makes it explicit what kind of assumptions about the data are being made when formulating the model. In this section we discuss the basic terminology and concepts of probabilistic modeling.

3.2.1 Random Variables and Probability Distributions

A central abstraction when operating under uncertainty is the **random variable**, that is, a variable whose value is subject to variations caused by randomness. In machine learning we observe a **sample**, that is some number of values of a random variable, and we would like to reason about the variable based on these observed values. Because of randomness we cannot know the particular value a random variable takes at any one time. Instead we attempt to express how likely it is for a particular value to occur by utilizing a **probability distribution**.

We employ the following notation. Random variables are written with capital Latin or Greek letters, for example X. Values that random variables take are written with lowercase Latin or Greek letters. Vectors, strings and lists are written in bold face. The set of values the random variable X takes is written as $\mathcal{X}$, and the set of values may be continuous or discrete. Samples are denoted by superscripts, such as $\mathbf{x}^{(i)}$ which denotes the ith sample value of the random variable $\mathbf{X}$.

The probability distribution can be defined in several ways. In this dissertation we employ exclusively the *probability density function*, denoted $p(X = x)$, that defines the relative likelihood for the random variable X to take the value x. We use the shorthand notation $p(x)$ if there is no ambiguity.

For a continuous probability density function it holds:

$$p(x) \geq 0 \tag{3.1}$$

$$\int_{\mathcal{X}} p(x)\, dx = 1 \tag{3.2}$$

For a discrete random variable, the integral in Expression (3.1) is replaced by a sum:

$$\sum_{x \in \mathcal{X}} p(x) = 1 \tag{3.3}$$

3.2.2 Parametric Models

A convenient way to define a probability distribution $p(x)$, is to use a **parametric model**, that is a family of probability functions $p(x|\theta)$ that define a probability distribution of the random variable X taking the value x given some parameter value θ. It is then assumed that each x is chosen randomly in a fashion that depends only on θ, and not, for example, on the previously generated x. This assumption is referred to as **independent, identically distributed** or **i.i.d.**

The parameters θ are typically estimated from some sample of values $\{x^{(i)}\}_{i=1}^{N}$. This can be performed in different fashions, and we will review different options in detail in Section 3.2.4. For now, we introduce one of the basic options, namely selecting the parameters such that the probability of the data is maximized given the parameters. Since $p(x|\theta)$ is known as the **likelihood**, this method is called **maximum likelihood** (ML):

$$\hat{\theta}_{ML} = \arg\max_{\theta} p(x|\theta) \tag{3.4}$$

Next, we review the parametric models employed in this dissertation.

The Geometric Distribution The geometric distribution is a discrete distribution defined on the natural numbers $0, 1, 2, \ldots$. It has a single parameter $\theta \in [0, 1]$, the probability of success. The geometric distribution can be interpreted as performing repeated Bernoulli-trial, that is tossing an unfair coin, until one gets the successful outcome on the xth trial. The output distribution is then defined over the number of required trials. The probability density function is given by:

$$p(x|\theta) = (1-\theta)^{x-1}\theta \tag{3.5}$$

The Gamma Distribution The Gamma distribution is a continuous distribution defined on the non-negative real axis. It has two parameters a and b. Its probability density function is generally asymmetric and it is bell-shaped if $a > 1$ and L-shaped otherwise. The probability density function is given by:

$$p(x|a, b) = \frac{b^a x^{a-1} e^{-bx}}{\Gamma(a)}, \tag{3.6}$$

where $\Gamma(a)$ is the Gamma-function $\Gamma(a) = \int_0^\infty u^{a-1}e^{-u}\mathrm{d}u$

The Categorical Distribution The categorical distribution is defined over some discrete event-space. For example, one may define a probability distribution over a set of words, and assume a we observe the sample $[the, the, a, the, car, the]$. For notational convenience, we assume that each event is mapped to its corresponding natural number j. The previous example can then be mapped to $[1, 1, 2, 1, 3, 1]$.

The categorical distribution employs a parameter vector $\boldsymbol{\theta}$ where the element θ_j defines the probability of event j:

$$p(X = j) = \theta_j \tag{3.7}$$

The probability of the sequence $\mathcal{U} = \{x^{(i)}\}_{i=1}^{N}$ is given by:

$$p(\mathcal{U}|\boldsymbol{\theta}) = \prod_{i=1}^{N} \theta_{x^{(i)}} = \prod_{j=1}^{K} \theta_j^{C(x^{(i)}=j)}, \tag{3.8}$$

where $C(a)$ denotes a function that counts the number of entries for which its argument function a is true, and K is the number of distinct events that can occur.

For the categorical distribution the maximum likelihood parameters are given by:

$$\hat{\theta}_j^{ML} = \frac{C(x^{(i)} = j)}{N} \tag{3.9}$$

The Multinomial Distribution Closely related to the categorical distribution is the multinomial distribution, and the two are at times conflated in the natural language processing literature. The only difference is that in the categorical distribution the observation sequence is important, whereas in the multinomial distribution the order does not matter, and only the resulting event counts are modeled. This requires the addition of a normalizing constant:

$$p(\mathcal{U}|\boldsymbol{\theta}) = \binom{N}{C(x=1)C(x=2)\dots C(x=K)} \prod_{j=1}^{K} \theta_j^{C(x^{(i)}=j)} \tag{3.10}$$

3.2.3 Probability Distributions of Several Random Variables

To model dependencies between several random variables $X_1, X_2, \dots, X_n$, we can define a probability distribution for the combined values of the variables, known as the **joint distribution** $p(x_1, x_2, \dots, x_n)$.

We can also define **conditional distributions** $p(x_1, \dots, x_k | x_{k+1}, \dots, x_n)$, that is the joint distribution of the variables $x_1, \dots, x_k$ given that we know the values of the variables $x_{k+1}, \dots, x_n$.

The relation between the joint and the conditional distributions is given by:

$$p(x_1, \dots, x_k | x_{k+1}, \dots, x_n) = \frac{p(x_1, \dots, x_n)}{p(x_k, \dots, x_n)} \tag{3.11}$$

The Product rule We can decompose a joint distribution into a sequence of products using the product rule. The product rule can be applied to any of the random variables X_i. The expressions are analogous for each variable. The factoring w.r.t. the variable X_1 is given by:

$$p(x_1, x_2, \dots, x_n) = p(x_1 | x_2, \dots, x_n) p(x_2, \dots, x_n) \tag{3.12}$$

The rule can be applied recursively, for example:

$$p(x_2, \dots, x_n) = p(x_2 | x_3, \dots, x_n) p(x_3, \dots, x_n) \tag{3.13}$$

The Sum rule We can **marginalize** away a variable X_k, by summing the joint distribution over all its possible values.

$$p(x_1, \ldots, x_{k-1}, x_{k+1}, \ldots, x_n) = \sum_{x_k} p(x_1, \ldots, x_n) \tag{3.14}$$

The distribution of a single variable $p(x_i)$ is called the **marginal distribution** of X_i, and it can be interpreted as the distribution for X_i when we do not know the values of the other variables.

Bayes' rule Central to probabilistic modeling is the ability to relate probability distributions from one variable to another. This is made possible by Bayes' rule:

$$p(x_1|x_2) = \frac{p(x_1)p(x_2|x_1)}{p(x_2)} \tag{3.15}$$

Independence If two random variables x_1 and x_2 are **independent**, then the value of one does not affect the other. Consequently the joint distribution can be factored as follows:

$$p(x_1, x_2) = p(x_1)p(x_2) \tag{3.16}$$

Often, independence holds only in a weaker form, such that if we know the value of some particular variable, then the remaining variables do not depend on one another. This is known as **conditional independence**. If x_1 and x_2 are conditionally independent given x_3 then:

$$p(x_1, x_2|x_3) = p(x_1|x_3)p(x_2|x_3) \tag{3.17}$$

3.2.4 Probabilistic Inference

Random variables for which we directly observe a sample are called **observed variables**. For example, words in a training data set. Variables that we do not observe are called **latent variables**, and their values must be reasoned about based on their relations to the observed variables. Probabilistic inference is based on utilizing the previously presented rules of probability distributions to infer what can be known about the variables we are interested in. In particular, we can also consider model parameters $\boldsymbol{\theta}$ as random variables with a probability distribution and therefore the rules of probabilistic inference can be applied to them as to any other random variable.

In a typical machine learning setting we are interested in taking some training data $\mathcal{D}$, learn generalizations from it that can be applied to some

new data point $\mathbf{x}^*$ which we have not observed in the data set, and compute some corresponding variable of interest $\mathbf{y}^*$. From the perspective of probabilistic inference we are therefore interested in $p(\mathbf{y}^*|\mathbf{x}^*, \mathcal{D})$. This distribution is the posterior distribution of $\mathbf{y}^*$, that is the distribution after having observed the training data. This formulation can be compared to the learning setups presented in Section 3.1. It is easy to see the commonalities with supervised learning, as $\mathbf{x}$ and $\mathbf{y}$ map directly to the input and output variables in the training set $\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^N$. In the case of unsupervised learning problems, the training set merely contains the input variables $\mathcal{U} = \{\mathbf{x}^{(i)}\}_{i=1}^N$. Therefore, the training set does not determine the target variable $\mathbf{y}$, but there are many valid possibilities. Many unsupervised problems can, nevertheless, also be considered as modeling $p(\mathbf{y}^*|\mathbf{x}^*, \mathcal{U})$. When there is a target task defined in advance, we can think of the task as defining the output variable $\mathbf{y}$. The morphological tasks described in Section 2.2 are examples of such tasks, where the correct morphological analysis takes the role of the variable $\mathbf{y}$.

Regardless of the learning setup being used there are a few typical assumptions employed in the probabilistic modeling. To model $p(\mathbf{y}^*|\mathbf{x}^*, \mathcal{D})$, the relations between the variables must be defined. A typical assumption is that $\mathbf{y}^*$ and $\mathbf{x}^*$ are produced by the same process that produced each sample in the training data, and that we can store that knowledge in a set of model parameters $\boldsymbol{\theta}$. In other words, we assume a conditional independence from the training data. We can then write:

$$p(\mathbf{y}^*|\mathbf{x}^*, \mathcal{D}) = \sum_{\Theta} p(\mathbf{y}^*, \boldsymbol{\theta}|\mathbf{x}^*, \mathcal{D}) \tag{3.18}$$

$$= \sum_{\Theta} p(\mathbf{y}^*|\boldsymbol{\theta}, \mathbf{x}^*, \mathcal{D}) p(\boldsymbol{\theta}|\mathbf{x}^*, \mathcal{D}) \tag{3.19}$$

$$= \sum_{\Theta} p(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}), \tag{3.20}$$

where the first step introduces the model parameters $\boldsymbol{\theta}$ by applying the sum rule, the second step applies the product rule, and finally the third step follows from conditional independence.

Moreover, typically the i.i.d. assumption is applied, that is each sample $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})^N$ is generated independently and is identically distributed: $p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{i=1}^N p(\mathbf{y}^{(i)}, \mathbf{x}^{(i)}|\boldsymbol{\theta})$. Similar reasoning can be employed to derive analogous expressions for the unsupervised case.

For some models, a simple form of $p(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\theta})$, can be found by analytic techniques. When this is not the case there are strategies for finding it approximately. A popular approach is based on drawing samples from the

distribution, which is often possible whether the distribution has a simple analytic form or not. Sampling methods can, however, be computationally demanding. An alternative approach is to assume that we do not need to marginalize over the space of model parameters Θ but that we should instead find a single value for the model parameters $\boldsymbol{\theta}$. This is referred to as finding a **point estimate** since we are approximating a distribution with a single point. The benefit is simpler and faster computational method while the downside is that the resulting estimate tends to underestimate the possible variability of the parameters. Since the choice of parameters $\boldsymbol{\theta}$ is crucial for the success of the point estimation approach, one strives to choose parameters that are optimal in some sense. In this approach, we will first calculate the optimal parameter $\boldsymbol{\theta}$ from $p(\boldsymbol{\theta}|\mathcal{D})$. This phase is referred to as **parameter estimation** or **training**. Then we can calculate values for $\mathbf{y}^*$ from $p(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\theta})$. This is generally referred to as **inference** or applying the model to new data. We can either calculate the full distribution or alternatively, just calculate the most probable value $\mathbf{y}^*$, that is

$$\hat{\mathbf{y}}^* = \arg\max_{\mathbf{y}' \in \mathcal{Y}} p(\mathbf{y}'|\mathbf{x}^*, \boldsymbol{\theta}) \tag{3.21}$$

The parameter estimation is based on the posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$. An application of Bayes' rule gives us the following:

$$\begin{aligned}
\hat{\boldsymbol{\theta}}_{OPT_{GEN}} &= \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta}|\mathcal{D}) && (3.22)\\
&= \arg\max_{\boldsymbol{\theta}} p(\mathcal{D}|\boldsymbol{\theta}) p(\boldsymbol{\theta}) && (3.23)\\
&= \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{N} p(\mathbf{y}^{(i)}, \mathbf{x}^{(i)}, \boldsymbol{\theta}) p(\boldsymbol{\theta}) && (3.24)
\end{aligned}$$

There are now two alternative ways to proceed. We can either build a **generative model**, which means that we choose an appropriate model for the **joint likelihood** $p(\mathbf{y}, \mathbf{x}|\boldsymbol{\theta})$. This approach can be used in both the supervised and unsupervised case. Alternatively, it can be noted that we can factor the expression further with the product rule:

$$\begin{aligned}
\hat{\boldsymbol{\theta}}_{OPT_{GEN}} &= \arg\max_{\boldsymbol{\theta}} p(\mathcal{D}|\boldsymbol{\theta}) p(\boldsymbol{\theta}) && (3.25)\\
&= \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{N} p(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}, \boldsymbol{\theta}) p(\mathbf{x}^{(i)}|\boldsymbol{\theta}) p(\boldsymbol{\theta}) && (3.26)\\
& && (3.27)
\end{aligned}$$

From this expression we can notice that we are solving two separate problems: First generating the input data via $p(\mathbf{x}|\boldsymbol{\theta})$ and then mapping

the input to the output with $p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta})$, that is the **conditional likelihood**. In many machine learning problems the input $\mathbf{x}$ is always observed, so modeling its generation is, in practice, unnecessary. Furthermore, if the input $p(\mathbf{x}|\boldsymbol{\theta})$ has complicated structure, modeling its generation may introduce unnecessary complexity. Instead, we may ignore $p(\mathbf{x}|\boldsymbol{\theta})$ and only focus on merely learning the conditional $p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta})$. This approach is known as a **discriminative modeling**, and it is applicable only in the supervised case, since we need observations of $\mathbf{y}$. The optimized expression is given by:

$$\hat{\boldsymbol{\theta}}_{OPT_{DISC}} = \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{N} p(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}, \boldsymbol{\theta}) p(\boldsymbol{\theta}) \tag{3.28}$$

The supervised learning setup is straightforward, whether generative or discriminative. In contrast, for unsupervised learning, the output variable $\mathbf{y}$ is an unobserved, latent variable. To proceed, more assumptions are required. One possibility is to formulate a generative model $p(\mathbf{y}, \mathbf{x}|\boldsymbol{\theta})$ and find the parameters based on the (marginal) likelihood

$$\hat{\boldsymbol{\theta}}_{OPT_{UNS}} = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta}|\mathcal{U}) = \arg\max_{\boldsymbol{\theta}} p(\mathcal{U}|\boldsymbol{\theta}) p(\boldsymbol{\theta}) \tag{3.29}$$

$$p(\mathcal{U}|\boldsymbol{\theta}) = \prod_{i=1}^{N} \sum_{\mathbf{y} \in \mathcal{Y}} p(\mathbf{y}, \mathbf{x}^{(i)}|\boldsymbol{\theta}) \tag{3.30}$$

The parameter prior distribution The distribution $p(\boldsymbol{\theta})$ appears in the expressions for supervised generative and discriminative learning, as well as unsupervised generative learning. It is known as the **prior** distribution for the parameters. It encodes degrees of belief in what the parameters are likely to be, before observing the training data, and can be used to guide the training based on prior knowledge. If we have no particular knowledge we can ignore the prior or use a non-informative prior that affects the result as little as possible. When seeking a point estimate we can either ignore the prior and only maximize the likelihood function or include the prior in the maximization. The former estimation technique is called **maximum likelihood** (ML) and the latter **maximum a posteriori** (MAP). Both ML and MAP estimation can be employed regardless of whether one performs generative or discriminative training, but the exact details of the procedures differ because generative and discriminative models estimate different distributions. The main drawback of maximum likelihood is that if the parametric model is too expressive compared to the number of samples available during parameter estimation, the maximum likelihood estimate will overfit. This tendency can be controlled by using a prior.

The Expectation-Maximization (EM) algorithm As described above, an unsupervised model that defines a joint distribution $p(\mathbf{y}, \mathbf{x}|\boldsymbol{\theta})$ can be trained by maximizing the marginal likelihood $p(\mathbf{x}|\boldsymbol{\theta})$. Such a model then has two latent variables, $\mathbf{Y}$ and $\boldsymbol{\theta}$. Since they depend on one another it is not straightforward to find the optimal parameters $\boldsymbol{\theta}$. The Expectation-Maximization algorithm is an iterative method that can be applied to models with this kind of interconnected latent variables, and it is applicable to both unsupervised learning and supervised learning with some missing data [Dempster et al., 1977, Bishop, 2006]. It can also be applied to semi-supervised learning by considering the annotation $\mathbf{y}^{(i)}$ for the unannotated data $\mathcal{U}$ as missing data. The EM algorithm finds a point estimate $\hat{\boldsymbol{\theta}}$ for the parameters that corresponds to a local maximum of the marginal likelihood $p(\mathcal{U}|\boldsymbol{\theta})$ or posterior $p(\boldsymbol{\theta}|\mathcal{U}) = p(\mathcal{U}|\boldsymbol{\theta})p(\boldsymbol{\theta})$. It can be derived for a model for which we have defined the joint distribution $p(\mathbf{x}, \mathbf{y}|\boldsymbol{\theta})$. The algorithm has two steps known as the Expectation step (E-step) and the Maximization step (M-step).

The E-step and M-step are repeated iteratively, updating the parameter value until convergence. We denote the parameter value at iteration n as $\hat{\boldsymbol{\theta}}_n$. In the E-step, one calculates the posterior probability of the latent variables for each training sample given the current model parameters $p(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}, \hat{\boldsymbol{\theta}}_n)$ In the M-step, one maximizes the expected joint likelihood or posterior, with the expectation taken over the distribution calculated in the E-step: $\hat{\boldsymbol{\theta}}^{(n+1)} = \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{N} \sum_{\mathbf{y}' \in \mathcal{Y}} p(\mathbf{x}^{(\mathbf{i})}, \mathbf{y}'|\boldsymbol{\theta}) p(\boldsymbol{\theta}) p(\mathbf{y}'|\mathbf{x}^{(\mathbf{i})}, \hat{\boldsymbol{\theta}}_n)$. The algorithm is written in pseudo code in Algorithm 1.

Algorithm 1 The Expectation-Maximization algorithm

Initialize $\hat{\boldsymbol{\theta}}_0$
while $p(\mathcal{U}|\boldsymbol{\theta})$ increases sufficiently **do**
  E-step: For each i, calculate $p(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}, \hat{\boldsymbol{\theta}}_n)$
  M-step: $\hat{\boldsymbol{\theta}}_{n+1} = \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{N} \sum_{\mathbf{y}' \in \mathcal{Y}} p(\mathbf{x}^{(\mathbf{i})}, \mathbf{y}'|\boldsymbol{\theta}) p(\boldsymbol{\theta}) p(\mathbf{y}'|\mathbf{x}^{(\mathbf{i})}, \hat{\boldsymbol{\theta}}_n)$
end while

3.3 Graphical Models

With the terminology defined so far we can describe more precisely how to handle uncertainty in practice. In general, we handle uncertainty by formulating a model that encodes the probabilistic dependencies by utilizing an appropriate joint or conditional distribution of the relevant variables.

Let $\mathbf{X} = X_1, X_2, \ldots, X_n$ and $\mathbf{Y} = Y_1, Y_2, \ldots, Y_n$ be a set of random variables that we refer to as the **input** and **output** variables, respectively. Let, the observed training data be denoted $\mathcal{D}$ and model parameters $\boldsymbol{\theta}$. Typically, the model parameters are latent and must be inferred from the observed data. In addition, the model may also contain additional latent variables $\boldsymbol{Z} = Z_1, Z_2, \ldots, Z_n$.

With a larger number of variables in the model, it becomes increasingly difficult to calculate. This is because the number of value combinations grow exponentially with the number of variables. For example, assuming for simplicity binary-valued variables, n variables can take 2^n possible values. Calculating over such large sets quickly becomes intractable. If, however, some of the variables are conditionally independent, efficient calculation may be possible.

Graphical models provide a framework for expressing the dependencies and independencies between random variables. The dependencies can be expressed as a graph, and this is the reason for the name graphical models. The framework enables the construction of arbitrarily large probabilistic models such that the same probabilistic inference principles can be applied.

There are two kinds of graphical models: directed and undirected ones. In both cases, the graph expresses how the joint distribution of the random variables is factored. The factorization implies a set of conditional independence conditions between the random variables.

3.3.1 Directed Models

Directed graphical models are based on factoring the joint distribution using the product rule, and then choosing appropriate expressions for the resulting conditional distributions. Examples of directed models include the widely used Hidden Markov Model which we will present next.

Hidden Markov Models

Hidden Markov Models (HMM) are directed graphical models of sequences, and they are used extensively in natural language processing [Rabiner, 1989, Manning and Schütze, 1999]. We focus here on the HMM with discrete observations. A HMM generates an observation sequence $\boldsymbol{x} = x_1, x_2, \ldots, x_n$ from a state sequence $\boldsymbol{y} = y_1, y_2, \ldots, y_n$. The state and observation sequences can be, for example, the part-of-speech tags and words in a sentence, respectively. Generally, all y_i take values from the same set

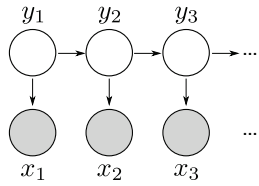

Figure 3.1. Directed graph corresponding to the Hidden Markov Model. White and gray circles denote latent and observed variables, respectively.

$y_i \in \mathcal{Y}$, and analogously $x_i \in \mathcal{X}$.

The joint distribution is factored such that the states are conditionally independent of everything given the previous state, and similarly the current observation only depends on the current state:

$$p(\boldsymbol{x}, \boldsymbol{y}) = \prod_{t=1}^{n} p(y_t|y_{t-1})p(x_t|y_t), \tag{3.31}$$

where we define a special start tag for y_0 such that $p(y_1|y_0) = p(y_1)$. The corresponding graph is shown in Figure 3.1 The distributions $p(y_t|y_{t-1})$ and $p(x_t|y_t)$ are separate categorical distributions for each value of y_{t-1} and y_t, respectively.

Parameter Estimation Parameter estimation in hidden Markov models can be solved analytically in a supervised setting where we observe $(\mathbf{x}, \mathbf{y})$-pairs. We can first note that $p(y_t|y_{t-1}) = p(y_t, y_{t-1})/p(y_{t-1})$ and $p(x_t|y_t) = p(x_t, y_t)/p(y_t)$. From the training data we get counts of the occurrences of each state and observation. Let the HMM be parameterized by $\boldsymbol{\theta} = \{\boldsymbol{\theta}_t, \boldsymbol{\theta}_e\}$, such that we utilize parametric models $p(y_t|y_{t-1}, \boldsymbol{\theta}_t)$ and $p(x_t|y_t, \boldsymbol{\theta}_e)$. We can compute the maximum likelihood parameters for the parametric models in question from the counts observed in the training data. For example, when utilizing a categorical distribution, its maximum likelihood is calculated with Expression (3.9).

In the unsupervised setting, the state sequence $\mathbf{y}$ is latent. Parameter estimation can be performed to find a local optimum of $p(\mathbf{x}|\boldsymbol{\theta})$ with the Expectation-Maximization algorithm (Section 3.2.4), which in this context is known as the Baum-Welch algorithm or the Forward-Backward algorithm [Baum et al., 1970, Rabiner, 1989, Manning and Schütze, 1999].

In the E-step we need to calculate the probability of the latent state sequence given the current parameters $\boldsymbol{\theta}^{(n)}$:

$$p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}^{(n)}) = \frac{p(\mathbf{y}, \mathbf{x}|\boldsymbol{\theta}^{(n)})}{p(\mathbf{x}|\boldsymbol{\theta}^{(n)})} = \frac{p(\mathbf{y}, \mathbf{x}|\boldsymbol{\theta}^{(n)})}{\sum_{\mathbf{y}} p(\mathbf{y}, \mathbf{x}|\boldsymbol{\theta}^{(n)})} \tag{3.32}$$

The M-step is then simply a question of updating the categorical distributions in the HMM given the distribution calculated in the E-step.

Calculating $p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}^{(n)})$ is complicated by the fact that the observation space $\mathbf{y}$ is exponential in size since each position can be occupied by one of the states in $\mathcal{Y}$. The expression can, however, be calculated efficiently with dynamic programming using the forward-backward factoring, as follows. Let $\mathbf{y}_{<1..t>}$ denote the values of the random variables $Y_1, Y_2, \ldots Y_t$. We can efficiently sum over all assignments to $\mathbf{y}_{<1..t>}$ with the assignment $Y_t = j$ if we know the corresponding sums at time $t-1$. This leads to the forward iteration:

$$\alpha_t(j) = \sum_{\mathbf{y}_{<1..t-1>}} p(\mathbf{y}_{<1..t-1>}, Y_t = j, \mathbf{x}_{<1..t>}|\boldsymbol{\theta}^{(n)}) \tag{3.33}$$

$$= \sum_{i \in \mathcal{Y}} p(Y_t = j|Y_{t-1} = i)p(x_t|Y_t = j)\alpha_{t-1}(i) \tag{3.34}$$

Similarly, for the backwards iteration we have:

$$\beta_t(i) = \sum_{\mathbf{y}_{<t+1..n>}} p(Y_t = i, \mathbf{y}_{<t+1..n>}, \mathbf{x}_{<t+1..n>}|\boldsymbol{\theta}^{(n)}) \tag{3.35}$$

$$= \sum_{j \in \mathcal{Y}} p(Y_{t+1} = j|Y_t = i)p(x_{t+1}|Y_{t+1} = j)\beta_{t+1}(j), \tag{3.36}$$

where the sequence is initialized with $\beta_n(i) = 1$.

Finally, the marginals of each state can be calculated as $p(Y_t = i, \mathbf{x}|\boldsymbol{\theta}^{(n)}) = \alpha_t(i)\beta_t(i)$. For the conditional in Expression (3.32) we get $p(\mathbf{x}|\boldsymbol{\theta}^{(n)})$ from either $\sum_{i \in \mathcal{Y}} \alpha_n(i)$ or $\beta_0(i)$.

Inference A common inference problem is to calculate the most probable assignment to $\mathbf{y}$ given an observation $\mathbf{x}$ and the model parameters $\boldsymbol{\theta}$:

$$\mathbf{y}^* = \arg\max_{\mathbf{y}'} p(\mathbf{y}'|\mathbf{x}, \boldsymbol{\theta}) = \arg\max_{\mathbf{y}'} \frac{p(\mathbf{y}', \mathbf{x}|\boldsymbol{\theta})}{p(\mathbf{x}|\boldsymbol{\theta})} \tag{3.37}$$

$$= \arg\max_{\mathbf{y}'} \prod_{t=1}^{n} p(y'_t|y'_{t-1})p(x_t|y'_t) \tag{3.38}$$

Since $\mathbf{x}$ does not vary, it can be ignored.

Here we can utilize an iteration similar to the forward-backward algorithm to efficiently calculate the maximum, since the best sequence that ends in state j at time t can be calculated by considering all states at position $t-1$ and the corresponding best sequence that ended there [Viterbi, 1967]. This iteration is known as the Viterbi-algorithm:

$$\delta_t(j) = \max_{i \in \mathcal{Y}} p(Y_t = j|p(Y_{t-1} = i)p(x_t|Y_t = j)\delta_{t-1}(i), \tag{3.39}$$

where $\delta_0(i) = 1$. After computing the δ we can recover the best assignment through a backward recursion:

$$\hat{y}_n^* = \arg\max_{i \in \mathcal{Y}} \delta_n(i)$$
$$\hat{y}_t^* = \arg\max_{i \in \mathcal{Y}} p(\hat{y}_{t+1}^* | Y_t = i) p(x_{t+1} | \hat{y}_{t+1}^*) \delta_t(i) \quad \text{for } t < n \tag{3.40}$$

3.3.2 Undirected Models

Undirected models decompose a joint or conditional distribution of variables $\mathbf{X} = X_1, X_2, \ldots, X_n$ into a product of factors $\Psi_a(\mathbf{X_a})$, where $\mathbf{X}_a$ denotes some subset of the random variables $\mathbf{X}$ and a is an integer index that varies $1 \ldots n_a$. Unlike the conditional probabilities employed in directed models, the factors are not normalized to sum to 1, but are only required to be non-negative. Normalization is performed globally. An undirected model can be written as:

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{a=1}^{n_a} \Psi_a(\mathbf{x}_a), \tag{3.41}$$

where Z is the normalizing factor that is required to make the expression sum to 1:

$$Z = \sum_{\mathbf{x}} \prod_{a=1}^{n_a} \Psi_a(\mathbf{x}_a) \tag{3.42}$$

Generally, calculating Z may be demanding as it may require summing over a set of values for $\mathbf{x}$ that is exponential in size, w.r.t. the number of random variables in the model.

Conditional Random Fields

Conditional Random Fields (CRF) are an undirected graphical model that relate the *input* variables $\mathbf{X}$ and the *output* variables $\mathbf{Y}$ which can both be structured variables, such as sequences, trees or graphs. Unlike generative models, CRFs model the conditional distribution $p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta})$, and do, therefore, not depend on the distribution of the input variables $p(\mathbf{x})$. Expression (3.42) for general undirected models then becomes:

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{a=1}^{n_a} \Psi_a(\mathbf{y}_a, \mathbf{x}_a) \tag{3.43}$$

It is typical to further assume that the $\log$ of the conditional likelihood is a linear function. We can then write the CRF as:

$$p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}) = \frac{1}{Z(\mathbf{x})} \prod_{a=1}^{n_a} \exp\{\sum_{k=1}^{k_a} \theta_{ak} f_{ak}(\mathbf{y}_a, \mathbf{x}_a)\}, \tag{3.44}$$

where for each factor a we have k_a **feature functions** f_{ak}. The feature functions can be chosen on a case by case basis. Typically, however, they are chosen such that there is a feature function that activates for each combination of the output variables $\mathbf{y}_a = \mathbf{y}'$, and a particular variant of the input variable $\mathbf{x}_a = \mathbf{x}'$. A common choice is an indicator function, such as:

$$f_{ak}(\mathbf{y}_a, \mathbf{x}_a) = \begin{cases} 1 & \text{if } \mathbf{y}_a = \mathbf{y}' \text{ and } \mathbf{x}_a = \mathbf{x}' \\ 0 & \text{otherwise} \end{cases} \tag{3.45}$$

However, the feature function may also return a continuous value. A benefit of the feature-function notation is that the parameters $\boldsymbol{\theta}$ can all be stored in a single parameter vector, rather than having separate storage for each combination of $\mathbf{y}_a$

We will consider concrete examples of CRFs including parameter estimation and inference, in Section 4.2 when we review the linear-chain CRF in detail.

3.4 Model Selection and Regularization

Many of the machine learning problems that we will encounter in this dissertation are ill-posed and, if naively formulated, lack unique optimal solutions. In particular, we must avoid **overfitting** which means that the model manages to fit the particular properties of the training sample very well, but does not generalize to unseen data. Model selection is a general term for diverse techniques that are used for choosing a model that will generalize well. An alternative approach is **regularization**, that is, modifying the objective function in such a way as to avoid known bad parameter values. We discussed utilizing prior probabilities to similar ends in Section 3.2.4. Regularization does often corresponds to formulating a prior probability over the space of possible parameter values, but this is not universally true. A general introduction to model selection and regularization can be found in [Alpaydin, 2004]

Next we review model selection with Minimum Description Length [Rissanen, 1989], and then we discuss L1 and L2 regularization.

3.4.1 Minimum Description Length (MDL)

The Minimum Description Length principle [Rissanen, 1987, 1989] argues that the useful information of a data set is achieved by a model that can maximally compress its **description length**. The description length

is a concept from information theory, related to how to communicate information over some channel and then reproduce it exactly at the other end [Shannon, 1948, McEliece, 2004]. Intuitively, one can think of MDL as a formalization of the idea of Occam's razor: Choose the simplest possible explanation for a given data set [Grünwald, 2005]. MDL then formalizes the notion of simplicity. MDL is closely related to the notion of Kolmogorov complexity [Kolmogorov, 1965] which defines the complexity of an object as the length of the minimal program that produces it as output. Kolmogorov complexity is, however, generally uncomputable while MDL is often efficiently computable.

MDL has been applied in several instances for unsupervised learning of morphology [Brent, 1993, de Marcken, 1996, Goldsmith, 2001, Creutz and Lagus, 2002, 2005a, 2007, Poon et al., 2009], usually by employing the two-part code [Grünwald, 2005]. The two-part code is characterized as a "crude" MDL by Grünwald [2005]. Since the later, more "refined" approaches have not been applied to learn morphology, we present here only the two-part code variant.

Minimum Description Length with Two-part Code

The goal of minimum description length with a two-part code is to choose the most appropriate model from a class of alternative models. MDL theory suggests that the best model $p(\mathbf{x}|\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ are the model parameters, is the one for which the combined description length of the model and the data given the model is minimized:

$$\boldsymbol{\theta}_{MDL} = \arg\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) + L(\mathbf{x}|\boldsymbol{\theta}) \tag{3.46}$$

We need to define the code-length functions $L()$ for both the model and the data given the model.

Following the presentation by Grünwald [2005], given a random variable $\mathbf{X}$ with the distribution $p(\mathbf{x})$ it is possible to construct a prefix-code whose code-length is $-\log p(\mathbf{x})$. Analogously, given some model $p(\mathbf{x}|\boldsymbol{\theta})$ with the parameters $\boldsymbol{\theta}$, we can encode observations of $\mathbf{X}$ with the coding-length $-\log p(\mathbf{x}|\boldsymbol{\theta})$. This latter expression assumes that both the sender and receiver know the model class and the parameters $\boldsymbol{\theta}$. Minimum description length theory suggests that when employing two-part codes one should metaphorically start by sending the encoded model, expressed with some appropriate code. The code-length of the model is given by $-\log p(\boldsymbol{\theta})$, where the distribution of the parameters $p(\boldsymbol{\theta})$ is formulated by the modeller, however, with the general aim of expressing the model

parameters as densely as possible. For example, encoding a categorical distribution (Section 3.2.2) is a question of encoding the number of distinct outcomes (and their labels if that is important) as well as the counts for each outcome. With these numbers transmitted the receiver can reconstruct the model and then employ it in decoding the data.

Relation of the Two-part Coding MDL and Maximum a Posteriori Estimation

To summarize the previous section we can see that the objective optimized in minimum description length with a two-part code is the combined code-length of the model and the data encoded with the model:

$$L(\boldsymbol{\theta}) = -\log p(\boldsymbol{\theta}) - \log p(\mathbf{x}|\boldsymbol{\theta}) \tag{3.47}$$

Minimizing the objective function in Expression (3.47) w.r.t. the parameters $\boldsymbol{\theta}$ is equivalent to maximizing $-\exp(L(\boldsymbol{\theta}))$, since the exponential function is monotonic. We can, therefore, see that minimizing Expression (3.47) above is equivalent to a maximum a posteriori (MAP) estimation problem:

$$\hat{\boldsymbol{\theta}}_{MAP} = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta})p(\mathbf{x}|\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta}|\mathbf{x}) \tag{3.48}$$

The second equivalence follows since by Bayes'rule $p(\boldsymbol{\theta}|\mathbf{x}) = \frac{p(\boldsymbol{\theta})p(\mathbf{x}|\boldsymbol{\theta})}{p(\mathbf{x})}$, and the normalization with $p(\mathbf{x})$ is unaffected by $\boldsymbol{\theta}$.

In other words, the two-part code minimum description length optimization problem is equivalent to a Maximum a Posteriori estimation problem with a particular prior $p(\boldsymbol{\theta})$ that is derived from a coding scheme for the model parameters $\boldsymbol{\theta}$.

3.4.2 Regularization

Regularization refers for a collection of techniques that control overfitting by favoring certain parameter values over others. The origins of regularization are in numerical problems but particular regularizers can also be given a probabilistic interpretation and be utilized in probabilistic models (see e.g. Tibshirani [1996]).

Regularization modifies the optimization problem of a parameter estimation method, such that if the unregularized method optimizes the function $l(\mathcal{D}, \boldsymbol{\theta})$, where $\mathcal{D}$ is a supervised or unsupervised data set. The regularized objective is then:

$$f(\mathcal{D}, \boldsymbol{\theta}) = l(\mathcal{D}, \boldsymbol{\theta}) + \lambda r(\boldsymbol{\theta}), \tag{3.49}$$

where λ is a scalar the controls the regularization cost, and $r(\boldsymbol{\theta})$ is a function of the parameters. Two common ones are L1 regularization, $r(\boldsymbol{\theta}) = \sum_j |\boldsymbol{\theta}_j|$, and L2 regularization, $r(\boldsymbol{\theta}) = \sum_j \boldsymbol{\theta}_j^2$. These regularizers can be given a probabilistic interpretation if $l(\mathcal{D}, \boldsymbol{\theta})$ is a likelihood function and $r()$ can be interpreted as defining a prior $p(\boldsymbol{\theta})$ over the parameter values. In that case, regularization corresponds to Maximum a Posteriori estimation. In particular, the L1 and L2 regularizers correspond to double exponential and Gaussian prior distributions, respectively [Tibshirani, 1996].

3.5 Levenshtein distance

Levenshtein distance is a metric that measures the distance between two strings s and t by the amount of edit operations needed to change one string into another [Levenshtein, 1966]. Originally, the strings considered were binary, but the algorithm can be generalized to arbitrary character sets.

Levenshtein distance can be calculated efficiently using a dynamic programming algorithm and the edit operations are insertion, deletion and substitution. The algorithm constructs a two-dimensional matrix C, such that:

$$C_{i,0} = i \tag{3.50}$$

$$C_{0,j} = j \tag{3.51}$$

$$C_{i,j} = \begin{cases} C_{i-1,j-1} & \text{if } s_i = t_j \\ 1 + min(C_{i-1,j}, C_{i,j-1}, C_{i-1,j-1}) & \text{otherwise} \end{cases}, \tag{3.52}$$

where the first two steps initialize the cost based on using only insertions or deletions, respectively. The final step then recurses over the strings and either notices that the strings are equivalent, and if they are not it chooses greedily the cheapest edit in the current position (see e.g. Navarro [2001], section 5.1.1 and 5.1.3).

It is straightforward to add a weight to the different operations. In the initialization steps, the cost of insertion and deletion are multiplied. In the recursion step, the addition with 1 is replaced with the cost of the locally least expensive operation.

4. Related Segmentation Methods

This section presents two central segmentation methods that will be applied or extended in the later chapters. First, we present Morfessor, an unsupervised method for morphological segmentation in Section 4.1. In Section 4.2 we then present linear-chain conditional random fields [Lafferty et al., 2001] for segmentation problems.

4.1 Morfessor

Morfessor is a family of methods for morphological segmentation. A comprehensive overview of all the methods within the same framework, as well as their coherent naming, is presented by Creutz and Lagus [2007]. A key property of Morfessor is the modeling of full concatenative morphology, in contrast to much of the previous unsupervised work that focused on only on affixing (see e.g. [Déjean, 1998, Schone and Jurafsky, 2000, Goldsmith, 2001]). For this reason, Morfessor is well suited to morphologically rich, highly agglutinative languages, such as Finnish or Turkish. Generally, the Morfessor methods employ generative probabilistic models that generate an observed corpus of words from a *morph lexicon*, that is a lexicon of stored morphological units. The methods produce the desired morphological segmentation by formulating the problem in such a way that the model generates the observed corpus through a latent morphological analysis variable that can trivially be converted into a segmentation. The Morfessor variants employ different parameter estimation methods; however, as is typical for latent variable models, they must all infer both a morphological analysis of the corpus and the model parameters.

In this dissertation we will work extensively with the earliest Morfessor variant, namely Morfessor Baseline [Creutz and Lagus, 2002] and, therefore, it will be the focus of the presentation in this chapter. We also em-

ploy Morfessor Categories-MAP [Creutz and Lagus, 2005a] as a reference method and for feature extraction. The other Morfessor variants, namely Baseline-Length [Creutz, 2003] and Categories-ML [Creutz and Lagus, 2004] are not employed in our work. Consequently, our presentation will begin in Section 4.1.1 from Morfessor Baseline, which will be presented in detail. Then, in Section 4.1.2 we will present Morfessor Categories-MAP by contrasting it to Morfessor Baseline.

Morfessor Baseline and Morfessor Categories-MAP are both formulated as a maximum a posteriori (MAP) optimization problem. Because of our focus on Morfessor Baseline, and to state explicitly which variables are observed and which are latent we will employ a different notation than Creutz and Lagus [2007]. We denote the observed training data consisting of words as $\mathcal{U} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$, the latent morphological analysis of each word string as $\mathbf{Z} = \{\mathbf{z}^{(i)}\}_{i=1}^{N}$, and the model parameters as $\boldsymbol{\theta}$. The training data words are strings, such that $\mathbf{x}^{(i)} \in \Sigma^*$. In this notation the problem can be formulated as follows. First, each training word is assumed to be generated independently:

$$p(\mathcal{U}, \mathbf{Z}, \boldsymbol{\theta}) = \prod_{i=1}^{N} p(\mathbf{x}^{(i)}, \mathbf{z}^{(i)}, \boldsymbol{\theta}) \tag{4.1}$$

Then, the joint distribution is factored as follows:

$$p(\mathbf{x}^{(i)}, \mathbf{z}^{(i)}, \boldsymbol{\theta}) = p(\mathbf{x}^{(i)}|\mathbf{z}^{(i)}, \boldsymbol{\theta})p(\mathbf{z}^{(i)})p(\boldsymbol{\theta}), \tag{4.2}$$

where we first factor using the product rule, and then assume that the latent analysis $\mathbf{z}$ and model parameters $\boldsymbol{\theta}$ are independent. Parameter estimation is then based on maximum a posteriori inference:

$$\hat{\boldsymbol{\theta}}_{MAP} = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta}|\mathcal{U}) \tag{4.3}$$

By Bayes' rule we get:

$$p(\boldsymbol{\theta}|\mathcal{U}) \propto p(\mathcal{U}|\boldsymbol{\theta})p(\boldsymbol{\theta}) \tag{4.4}$$

It can be noted that the normalization is irrelevant for the maximization in Expression (4.3).

4.1.1 Morfessor Baseline

In this section we discuss Morfessor Baseline [Creutz and Lagus, 2002, 2007]. The presentation follows Creutz and Lagus [2007], since some model components were improved upon and some of the details missing in the original publication [Creutz and Lagus, 2002].

The Morfessor Baseline model consists of a single categorical distribution[1] over morphs $P(\mathbf{m}|\boldsymbol{\theta})$, where $\mathbf{m}$ and $\boldsymbol{\theta}$ denote a morph string and the parameters defining the probabilities for each morph, respectively. In Morfessor Baseline, the latent morphological analysis $\mathbf{z}^{(i)}$ of each word $\mathbf{x}^{(i)}$ is simply a segmentation. For example, for the word $\mathbf{x}^{(i)} =$ *'coffee'*, the corresponding variable $\mathbf{z}^{(i)}$ ranges over all segmentations, encoded as a list of substrings of arbitrary length $\mathbf{z}^{(i)}$ whose concatenation produces $\mathbf{x}^{(i)}$, where $n = |\mathbf{z}^{(i)}| \geq 1$:

$$\mathbf{x}^{(i)} = \mathbf{z}_1^{(i)} \circ \mathbf{z}_2^{(i)} \circ \cdots \circ \mathbf{z}_n^{(i)}, \tag{4.5}$$

where $\circ$ denotes concatenation. Example segmentations include, [*'cof', 'fe', 'e'*], [*'coffee'*], and [*'c', 'offee'*].[2]

Morfessor Baseline Likelihood

The likelihood assumes each word $\mathbf{x}^{(i)} \in \mathcal{U}$ is generated independently.

$$p(\mathcal{U}|\boldsymbol{\theta}) = \prod_{i=1}^{N} p(\mathbf{x}^{(i)}|\boldsymbol{\theta}) \tag{4.6}$$

Moreover, given a word $\mathbf{x}^{(i)}$ with a segmentation $\mathbf{z}^{(i)}$, the likelihood assumes that each morph $\mathbf{z}_j^{(i)}$ is generated independently. However, the segmentation $\mathbf{z}^{(i)}$ is a latent variable and, therefore, needs to be marginalized over to get the (marginal) likelihood:

$$p(\mathbf{x}^{(i)}|\boldsymbol{\theta}) = \textstyle\sum_{\mathbf{z}^{(i)} \in \mathrm{SEG}(\mathbf{x}^{(i)})} p(\mathbf{x}^{(i)}|\boldsymbol{\theta}, \mathbf{z}^{(i)}) p(\mathbf{z}^{(i)}) \tag{4.7}$$

$$p(\mathbf{x}^{(i)}|\boldsymbol{\theta}, \mathbf{z}^{(i)}) = \textstyle\prod_{j=1}^{|\mathbf{z}^{(i)}|} p(\mathbf{M} = \mathbf{z}_j^{(i)}|\boldsymbol{\theta}), \tag{4.8}$$

where $\mathrm{SEG}(\mathbf{x}^{(i)})$ denotes the set of possible segmentations for the word $\mathbf{x}^{(i)}$. Generally, a uniform distribution is assumed for $p(\mathbf{z})$. Consequently, it will not affect subsequent optimization. In the next section we will specify the details of the model parameters $\boldsymbol{\theta}$. Here, it is sufficient to state that $\boldsymbol{\theta}$ contains the parameters necessary to encode the categorical distribution $p(\mathbf{m}|\boldsymbol{\theta})$ over the set of all strings $\mathbf{m} \in \Sigma^*$.

[1] Several of the publications refer to the distribution as multinomial. This is approximately true, but since the multinomial coefficient is not employed, strictly speaking, a categorical distribution is employed

[2] For comparison to the notation in [Creutz and Lagus, 2007]: our segmentation variable $\mathbf{z}^{(i)}$ defines the sequence of morphs $\mathbf{z}_j^{(i)}$ corresponding to the original μ_i, the string value of the variable $\mathbf{z}_j^{(i)}$ corresponds to $form(\mu_i)$, and our $\boldsymbol{\theta}$ has no exact correspondence but contains $form(\mu_i)$ as well as a parameter vector that can be calculated from $freq(\mu_i)$.

It can be noted that Expression (4.8) is non-normalized, as the varying number of morphs $|\mathbf{z}^{(i)}|$ in each analysis $\mathbf{z}^{(i)}$ needs to be modeled for the expression to sum to 1 (this is also pointed out by Virpioja [2012, Section 6.4.1.4]). In practice, this non-normalized expression has been employed in Morfessor Baseline implementations.

Morfessor Baseline Prior

The Morfessor Baseline prior distribution is derived from the minimum description length principle by utilizing the two-part code approach described in Section 3.4.1. Intuitively, the prior is defined based on the idea that one needs to transfer the observed corpus $\mathcal{U}$ as compactly as possible to a receiver. To enable compression one first transmits a model $p(\mathbf{x}|\boldsymbol{\theta})$ to the receiver and then one transmits the data $\mathcal{U} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$ encoded with the model. In practice, we are only interested in the code length and it is defined by negative logarithm of a distribution $p(\boldsymbol{\theta})$. In two-part code MDL the definition of $p(\boldsymbol{\theta})$ is for the modeller to decide, but it needs to be defined in such a way that all the necessary parameters are included and to produce as compact a code for the set of parameters as possible.

In the previous section we merely stated that $\boldsymbol{\theta}$ defines the distribution $p(\mathbf{m}|\boldsymbol{\theta})$ over the set of all strings. However, this set is infinite and, therefore, not compact as well as impractical to work with. In Morfessor it is assumed that $p(\mathbf{m}|\boldsymbol{\theta})$ is *sparse*, such that for many substrings s it holds that $p(\mathbf{m} = s|\boldsymbol{\theta}) = 0$. Let $\mathbf{m}$ be the set of substrings s for which $p(\mathbf{m} = s|\boldsymbol{\theta}) > 0$. These are referred to as *stored* morphs, and together with their probability weight they form the *morph-lexicon*. Consequently, the distribution can be represented by listing the set of substrings $\mathbf{m}_k \in \mathcal{M}$ with a nonzero probability and their probability weight ϕ_k. Then the model parameters can be defined as $\boldsymbol{\theta} = \{\mathcal{M}, \boldsymbol{\phi}\}$. The categorical distribution is given by:

$$p(\mathbf{m} = s|\boldsymbol{\theta}) = \begin{cases} \phi_k & \text{if } \exists \mathbf{m}_k \in \mathcal{M} \text{ s.t. } \mathbf{m}_k = s \\ 0 & \text{otherwise} \end{cases} \tag{4.9}$$

It follows that the model becomes variable-length, depending on the number of morphs with nonzero probability. We now proceed to discuss the details of the prior. To transfer the model we need to encode:

1. The number of morphs $M = |\mathcal{M}|$, because the model is variable-length

2. For each stored morph $\mathbf{m}_k$, its string

3. The probability values ϕ_k of each stored morph $\mathbf{m}_k$

For all these we must define a probability distribution from which we can derive the code length as the negative logarithm of the probability. Morfessor Baseline employs the following expression:

$$p(\boldsymbol{\theta}) = p(M)p(\boldsymbol{\phi}) \prod_{k=1}^{M} p(\mathbf{m}_k|c,l)M!, \tag{4.10}$$

where it is assumed that each morph string is encoded independently, whereas the probability weights $\boldsymbol{\phi}$ are encoded jointly. The factorial of M compensates for the fact that the same lexicon can be encoded in arbitrary order, and the number of such orders is $M!$, and an efficient code can utilize this fact. The effect of encoding the number of morphs M is very small, so in practice it is ignored [Creutz and Lagus, 2007, Chapter 3.2]. The morph strings are transferred by first transferring the length of the morph in characters, and then the characters themselves according to the probability distribution:

$$p(\mathbf{m}_k|\boldsymbol{c},l) = p(l = |\mathbf{m}_k|) \prod_{i=1}^{|\mathbf{m}_k|} p(c = \mathbf{m}_{ki}), \tag{4.11}$$

where the length-distribution l and character probabilities c are hyperparameters that are thought to be known. The length-distribution $p(l = |\mathbf{m}_k|)$ can either be defined explicitly, for example by applying a gamma distribution, or implicitly by adding an end of morph-character and generate it from $p(c)$ [Creutz, 2003, Creutz and Lagus, 2005b]. The parameters of $p(c)$ are in practice estimated from the empirical character distribution in the training set.

Next, the probability weights ϕ_k of each morph must be transferred. Any distribution over the space of real numbers could be employed. Morfessor takes an alternative approach. Firstly, this is because formulating a distribution over real numbers is non-trivial.[3] Secondly, a distribution over real numbers does not utilize the specific properties of categorical probability weigths $0 \leq \phi_k \leq 1$ and $\sum_{k=1}^{K} \phi_k = 1$ to produce a maximally compact code. In addition to utilizing these properties of ϕ_k, Morfessor employs the further constraint that the probability weights $\boldsymbol{\phi}$ will always be Maximum Likelihood parameters corresponding to some latent segmentation $\{\mathbf{z}^{(i)}\}_{i=1}^{N}$. For a categorical distribution the Maximum Likelihood parameters are given by the count n_k/N_k for each event k, where

[3] Rissanen's universal for positive numbers is one well-known alternative [Rissanen, 1989]

$N_k = \sum_{k=1}^{K} n_k$. Note, that the counts of each morph can simply be counted from a known $\{\mathbf{z}^{(i)}\}_{i=1}^{N}$. Therefore, a prior distribution can be defined for such counts rather than for the vector of real event probabilities $\boldsymbol{\phi}$. The number of combinations of M_k numbers that add up to N_k is $\binom{M_k-1}{N_k-1}$. We can thus define a distribution for the counts n_k as:

$$p(\boldsymbol{\phi}) = p(n_1, \ldots, n_M | M_k, N_k) = 1/\binom{N_k - 1}{M_k - 1} \tag{4.12}$$

Finally, we discuss some properties of the presented prior. The effect of this prior intuitively is to penalize lexicons that store many and long strings. In practice, the generation of morph strings from $p(\mathbf{m}_k|c, l)$ has the largest effect on the objective function, thus favoring small lexicons with short morphs. Similarly, coding of the probability weights $\boldsymbol{\phi}$ increases with growing morph token count N_k. The effect of the morph lexicon size M_k is non-trivial. Expression (4.12) favors either small values of M_k, that is close to 1; or large values of M_k, that is close to N_k. In summary, the prior will assign a penalty for adding morphs with nonzero probability to the model – adding parameters to the model and this penalty is most strongly affected by the string length of the added morph.

Parameter Estimation

As presented in the previous section we have a maximum a posteriori estimation problem for a latent variable model with a prior that promotes simplicity in the model. Generally, the standard choice for estimating the MAP model of a latent variable model would be the Expectation-Maximization (EM) algorithm (Section 3.2.4). For Morfessor Baseline, EM cannot be applied in a straightforward fashion because the MDL-based prior is non-continuous. In particular, the prior is only affected by whether the probability $p(\mathbf{m} = \mathrm{s}|\boldsymbol{\theta})$ of a morph s is nonzero or exactly zero.

Consequently, Morfessor Baseline employs a heuristic training algorithm that greedily optimizes an approximation of the MAP objective. The algorithm is shown as pseudocode in Algorithm 2, which is a more abstract presentation of the one provided in [Creutz and Lagus, 2005b]. Although not discussed in the original publications [Creutz and Lagus, 2002, 2007], it is evident that this parameter estimation procedure does not optimize the MAP objective in Expression (4.3) exactly. Instead, it optimizes a related objective function that is more amenable to local search. The optimized objective function is based on the simplification that all probability mass is focused in a single segmentation for each word $\hat{\mathbf{Z}} = \{\hat{\mathbf{z}}^{(i)}\}_{i=1}^{N}$.

Given this assumption, the summing over all latent segmentations in Expression (4.8) reduces to merely summing over the single segmentation $\hat{\mathbf{Z}}$, and (4.8) reduces to:

$$\hat{\boldsymbol{\theta}}_{BL} = \arg\max_{\boldsymbol{\theta},\hat{\mathbf{Z}}} \prod_{i=1}^{N} p(\mathbf{x}^{(i)}|\boldsymbol{\theta}, \hat{\mathbf{z}}^{(i)}) p(\boldsymbol{\theta}) \tag{4.13}$$

This objective is then optimized in an iterative fashion. For each word $\mathbf{x}^{(i)}$ different segmentations $\hat{\mathbf{z}}^{(i)}$ are attempted, and for each value of the $\hat{\mathbf{z}}^{(i)}$ the parameters $\boldsymbol{\theta}$ are updated to their ML estimate given the current segmentation $\hat{\mathbf{Z}}$. Since $\hat{\mathbf{z}}^{(i)}$ defines a segmentation of the input words $\mathbf{x}^{(i)}$, we can calculate the ML estimate for $\boldsymbol{\theta}$ simply by counting the number of occurrences of each morph $\mathbf{m}$ in the segmentation $\hat{\mathbf{z}}^{(i)}$. The local search procedure chooses the segmentation $\mathbf{z}^{(i)}$ such that it results in the highest posterior probability and then the same procedure is applied to the next word. All words are processed in random order. The procedure is repeated for all words until the posterior probability given $\hat{\mathbf{Z}}$, that is, $\prod_{i=1}^{N} p(\mathbf{x}^{(i)}|\hat{\boldsymbol{\theta}}, \hat{\mathbf{z}}^{(i)}) p(\hat{\boldsymbol{\theta}})$ grows less than a pre-set threshold.

The training algorithm employs a parameter-binding approach that is useful to avoid getting stuck in bad local optima. As stated previously, the prior probability is only affected when the probability of a morph $\mathbf{m}$ is set to exactly zero. This requires that the segmentation $\hat{\mathbf{Z}}$ does not contain any instance of $\mathbf{m}$. Discovering such analyses by processing one word at a time is inefficient. Therefore, the analyses are bound together with a hierarchical decomposition. Each word is either not split at all, or is split in two parts. The two parts are then recursively either not split, or split in two smaller parts, until further splits no longer produce a better segmentation. During parameter estimation the segmentations $\hat{\mathbf{z}}^{(i)}$ are bound in such a way that when modifying the analysis of morph $\mathbf{m}_k$ in word $\mathbf{x}^{(i)}$ all other analyses that contain $\mathbf{m}_k$ *at any level* of the hierarchical decomposition are modified simultaneously. Consider, for example a training set $\mathcal{U}$ = [*'woodworker', 'woodworkers', 'wood'*], and a current segmentation where we have only split the second word $\hat{\mathbf{Z}}$ = [[*'woodworker'*], [*'woodworker', 's'*], [*'wood'*]]. It is easy to see that such a segmentation can be better than a no-split segmentation, as it allows for elimination of the string *'woodworkers'* from the morph lexicon, and thereby increasing the prior probability of the parameter values $\hat{\boldsymbol{\theta}}$. In contrast, if we independently split up the first word into [*'wood', 'worker'*], then while we can reuse the morph *'wood'* in two forms we, nevertheless, need to introduce the previously unneeded morph *'worker'* into the lexicon. Depending on

the exact word lengths and current model parameters, this may or may not be a better analysis. Note, however, that when employing the parameter binding we would simultaneously split up also the other the instance of *'woodworker'* in the second word. This latter approach allows for the elimination of the morph *'woodworker'* from the lexicon, and therefore the prior probability is improved. This parameter binding is achieved in Algorithm 2 by the function call update_segmentations($\hat{\mathbf{Z}}, \mathbf{m} \leftarrow \hat{\mathbf{z}}_{mk}$) which is interpreted such that the current segmentation of all words $\hat{\mathbf{Z}}$ is modified by replacing all instances of the morph $\mathbf{m}$ with $\hat{\mathbf{z}}_{mk}$ at any level of the hierarchy.

The notation $\mathbf{m}_{<k..l>}$ denotes the substring of $\mathbf{m}$ from character index k to l.

Algorithm 2 The learning algorithm

Initialize segmentation $\hat{\mathbf{Z}} = \{\hat{\mathbf{z}}^{(i)}\}_{i=1}^{N}$ and model parameters $\hat{\boldsymbol{\theta}}$
while $\prod_{i=1}^{N} p(\mathbf{x}^{(i)}|\hat{\boldsymbol{\theta}}, \hat{\mathbf{z}}^{(i)})p(\hat{\boldsymbol{\theta}})$ increases sufficiently **do**
  for $\mathbf{x}^{(i)} \in \mathcal{U}$ in random order **do** optimize($\mathbf{x}^{(i)}$)
end while
function optimize($\mathbf{m}$)
  $\mathcal{Z}_m \leftarrow \left[\mathbf{m}\right] \cup \left[(\mathbf{m}_{<1..l>}, \mathbf{m}_{<(l+1)..|\mathbf{m}|>}) : l \in 1, ..., |\mathbf{m}| - 1\right]$
  for $k = 1 \ldots |\mathcal{Z}_m|$
    let $\hat{\mathbf{z}}_{mk}$ be the kth alternative segmentation in $\mathcal{Z}_m$
    $\hat{\mathbf{Z}}_{mk} \leftarrow$ update_segmentations($\hat{\mathbf{Z}}, \mathbf{m} \leftarrow \hat{\mathbf{z}}_{mk}$)
    $\hat{\boldsymbol{\theta}}_{\hat{\mathbf{Z}}_{mk}} \leftarrow \arg\max_{\boldsymbol{\theta}} p(\mathcal{U}|\boldsymbol{\theta}, \hat{\mathbf{Z}}_{mk})$ (ML estimate for $\boldsymbol{\theta}$ given $\hat{\mathbf{Z}}_{mk}$)
  $k_{best} \leftarrow \arg\max_{k=1...|\mathcal{Z}_m|} \prod_{i=1} p(\mathbf{x}^{(i)}|\hat{\boldsymbol{\theta}}_{\hat{\mathbf{Z}}_{mk}}, \hat{\mathbf{z}}^{(i)}_{mk})p(\hat{\boldsymbol{\theta}}_{\hat{\mathbf{Z}}_{mk}})$
  $\hat{\mathbf{Z}} \leftarrow \hat{\mathbf{Z}}_{k_{best}}$
  $\hat{\boldsymbol{\theta}} \leftarrow \hat{\boldsymbol{\theta}}_{\hat{\mathbf{Z}}_{mk_{best}}}$
  if $\hat{\mathbf{z}}_{mk_{best}}$ involved a split at l **then** optimize($\mathbf{m}_{<1..l>}$); optimize($\mathbf{m}_{<(l+1)..|\mathbf{m}|>}$)

As the algorithm considers, in the worst case, $|\mathbf{x}^{(i)}| - 1$ potential splits for each form, its worst case time complexity per batch is $O(E_{(i)}\{|\mathbf{x}^{(i)}|\}^2 N)$. The worst case is realized when each subsequent split for each word occurs at either end of the word, such that recursive reanalysis is applied to a morph whose length is only one character shorter than at the previous level. The number of batch passes over the data is typically between 3 and 11.

Inference

Subsequent to training, the model can be applied to analyze new word forms x^*. By utilizing a variant of the *Viterbi algorithm*, we can find the most probable segmentation for a word efficiently by utilizing dynamic programming.

$$\hat{\mathbf{z}}^* = \underset{\mathbf{z}\in\mathrm{SEG}(\mathrm{x}^*)}{\arg\max}\ p(\mathbf{z}|\mathrm{x}^*,\boldsymbol{\theta}) = \underset{\mathbf{z}\in\mathrm{SEG}(\mathrm{x}^*)}{\arg\max}\ p(\mathrm{x}^*|\mathbf{z},\boldsymbol{\theta}), \tag{4.14}$$

where the latter equivalence is derived using Bayes' rule: $p(\mathbf{z}|\mathrm{x}^*,\boldsymbol{\theta}) = p(\mathrm{x}^*|\mathbf{z},\boldsymbol{\theta})p(\mathrm{x}^*)/p(\mathbf{z})$, and $p(\mathrm{x}^*)$ does not vary with $\mathbf{z}$, and we assume $p(\mathbf{z})$ is non-informative (uniform).

For inference, the Morfessor Baseline model is equivalent with a Hidden Markov Model where each state (morph) has a variable length. Such a model is a special case of a semi-Markov model (see e.g. [Yu, 2010]). Here, the observation is the sequence of $|\mathrm{x}^*|$ letters that form the word x^* and the states are the morphemes of the word. As the states can emit observations of different lengths, a grid $\boldsymbol{g}$ of length $|\mathrm{x}^*|$ is required to fill with the best unnormalized probability values $\alpha(\boldsymbol{g}_i)$ and paths. As the morphs are generated independently, the model is 0th order semi-Markov model and the grid is a one-dimensional table. The grid position $\boldsymbol{g}_i$ indicates that the first i letters are observed. At each time step, we proceed with one letter and insert the value $\alpha(\boldsymbol{g}_l) = \max_k \alpha(\boldsymbol{g}_k)P(\mathbf{m} = \mathrm{x}^*_{<k..l>}|\boldsymbol{\theta})$ and path indicator $\psi(\boldsymbol{g}_l) = \arg\max_k \alpha(\boldsymbol{g}_k)P(\mathbf{m} = \mathrm{x}^*_{<k..l>}|\boldsymbol{\theta})$ to the grid. We can arrive at $\boldsymbol{g}_l$ from any of the positions $\boldsymbol{g}_k$ between $\boldsymbol{g}_1$ and $\boldsymbol{g}_{l-1}$: The letters between k and l form the next morph $\mathrm{x}^*_{<k..l>}$. For the substrings that are not part of the lexicon, we allow for adding them by assigning a probability comparable to what would result from adding them into the lexicon. The length of the final grid position is, therefore, $|\mathrm{x}^*|$, and consequently the resulting time complexity is $O(|\mathrm{x}^*|^2)$ for the algorithm.

Discussion

The benefit of the heuristic algorithm is that it optimizes the likelihood and prior part of the objective function in parallel. The downside is that although the algorithm obviously converges to a local optimum of the modified objective function in Expression (4.13) it has not been shown how closely related this is to a local optimum of the original MAP objective in Expression (4.3). Intuitively, with the prior chosen, words tend to have few segmentations with high probability, and therefore, the approximation may be fairly close to the original objective. However, regardless of the intrinsic properties of the approximation, the performance in the

actual morphological segmentation task has, nevertheless, been demonstrated empirically in the original publications [Creutz and Lagus, 2002, 2007].

A key aspect of the Morfessor Baseline method are the roles played by the prior and the likelihood in the decision to segment or not to segment. Generally, it can be seen that the likelihood prefers to store whole words rather than splitting them up, as the pieces need to be generated independently, and that is costly in general. In contrast, the prior prefers to store as few morphs as possible, especially longer ones. The method will segment to an amount that is a compromise between these two factors.

We can also make a connection between Morfessor Baseline and the classical method of Harris [1955] for morphological segmentation, based on local maxima in letter successor variety. On an abstract level this corresponds to splitting words at positions of high uncertainty. If we consider the Morfessor Baseline method we can see that we ask the method to find a sequence of independent morphs and force some amount of splitting by assigning a cost for each stored morph. Despite looking different on the surface, it can be noted that this is essentially a latent variable version of a very similar idea. Instead of finding positions of high uncertainty we ask for morphs with strong internal dependence, because morphs are forced to be generated independently, and therefore the only remaining strategy for modeling dependence is to avoid segmenting strings with high internal dependence.

A further connection can be made to the morphological segmentation methods in the adaptor grammar framework (see e.g [Goldwater, 2006, Johnson et al., 2007]). Adaptor grammars define a two-step process for the generation of discrete variables. The first step, known as the generator, produces the entities under study - words, morphemes, syllables or some other suitable unit. The second step, the adaptor, then produces a power-law distribution over the units by either generating a new sample from the generator, or generating a previously generated entity with a probability proportional to how many times said entity was produced in the past. The correspondence is not exact mathematically, but the units stored in the adaptor are similar to the stored morphs in the Morfessor Baseline lexicon, and the generator is similar to the generation of the units from letters in the Morfessor Baseline prior. Unlike Morfessor, however, the adapted units can be easily combined into probabilistic grammars, and there is no requirement that the generator needs to operate on

letters, but a wide range of probabilistic processes can be utilized. Consequently, adaptor grammar models are typically more detailed than those employed in the Morfessor-framework. This flexibility is enabled by utilizing general inference procedures, particularly Bayesian model averaging techniques, such as Gibbs sampling. The general inference algorithms can be adapted to different models without requiring parameter estimation and inference procedures to be redeveloped completely.

4.1.2 Morfessor Categories-MAP

In this section we review Morfessor Categories-MAP and contrast it to Morfessor Baseline [Creutz and Lagus, 2002, 2007]. Morfessor Categories-ML Creutz and Lagus [2004] and Categories-MAP [Creutz and Lagus, 2005a] differ from Baseline in several fashions. Firstly, rather than the unlabeled segmentation of Morfessor Baseline, the latent analysis $\mathbf{z}^{(i)}$ is a labeled segmentation where each segment belongs to one of three morphotactic categories: prefix, stem, and suffix (tagged as PRE, STM, SUF, respectively). For example consider the analysis of *unavoidable*:

PRE	STM	SUF
un	avoid	able

The observed words are produced by a hidden Markov model with states corresponding to these categories. Unlike Morfessor Baseline, in Categories-ML the model complexity is controlled heuristically. In contrast, and similarly to Morfessor Baseline, Categories-MAP employs a formulation where the model complexity is managed by employing a minimum description length prior over a morph lexicon. However, the details of this lexicon differ from Morfessor Baseline. Whereas Morfessor Baseline employs a hierarchical decomposition of the training words only for parameter-binding during parameter estimation, Morfessor Categories-MAP employs hierarchical decomposition also in the prior distribution, that is a morph can be stored by referring to smaller morphs.

Finally, the parameter estimation method employed is different from Morfessor Baseline. Similarly to the greedy optimization method presented in Section 4.1.1 Categories-MAP employs a heuristic algorithm which optimizes the objective function utilizing a single analysis $\hat{\mathbf{Z}}$ of the training data. There are several differences between the training algorithms, which we will, however, not discuss in more detail as they are not relevant for the purposes of this dissertation.

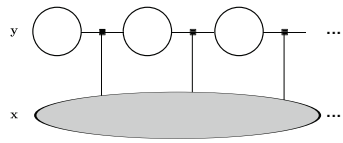

Figure 4.1. Factor graph corresponding to the linear-chain CRF Model. White and gray circles denote output and input variable, respectively

4.2 Segmentation with Linear-Chain Conditional Random Fields

In this section we present linear-chain conditional random fields (CRF), a framework of probabilistic models for segmentation and sequence labeling. We will later apply linear-chain CRFs to morphological segmentation in Section 6.5.

As presented in Section 3.3.2, conditional random fields are a framework for conditional probabilistic models where a set of task-specific feature functions are used to define a conditional probability distribution between structured input and output variables.

Linear-chain CRFs are conditional models of sequences, and the *output* variables $\mathbf{Y}$ form a linear-chain, similarly to the state sequence in a hidden Markov model (Section 3.3.1). The factor-graph for the linear-chain CRF is shown in Figure 4.1. As the dependencies between the input variables $\mathbf{x}$ are not modeled by the CRF, the factors can depend on any position in the input. The relation to HMM is not only superficial, but it turns out that the inference algorithms used for the HMM, the Viterbi algorithm and the Forward-Backward iteration, are applicable also for the linear-chain CRF.

The linear-chain CRF models the dependencies between the label sequence $\mathbf{y} = (y_1, y_2, \ldots, y_T)$ and an input sequence $\mathbf{x} = (x_1, x_2, \ldots, x_T)$. Each output variable y_t takes values from the same set $y_t \in \mathcal{Y}$; analogously $x_t \in \mathcal{X}$. Consequently, Expression (3.44) for the general CRF simplifies into:

$$p\left(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}\right) = \frac{1}{Z(\mathbf{x})} \prod_{t=2}^{|\mathbf{x}|} \exp \sum_{k=1}^{K} \left({\boldsymbol{\theta}_k}^{\top} f_k(y_{t-1}, y_t, \mathbf{x}, t) \right), \tag{4.15}$$

$$Z(\mathbf{x}) = \sum_{\mathbf{y}} \prod_{t=2}^{|\mathbf{x}|} \exp \sum_{k=1}^{K} \left({\boldsymbol{\theta}_k}^{\top} f_k(y_{t-1}, y_t, \mathbf{x}, t) \right), \tag{4.16}$$

where t indexes the position in the sequence, $\boldsymbol{\theta}$ denotes the model parameter vector, and f_k is a vector-valued feature function. The model parameter vector $\boldsymbol{\theta}$ is estimated discriminatively based on a training set

of exemplar input-output pairs $\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{N}$.

In order to perform segmentation the label set $\mathcal{Y}$ must be chosen, such that the labeling specifies a segmentation. There are several possible label sets that can be employed, as long as they are isomorphic to a segmentation. This is necessary, since we must be able to transform the segmentations in the annotated training data into the sequence-labeling format, and vice versa when applying the method to new data.

The minimal labeling choice for segmentation is marking the beginning of segment *B* and the middle of segment *M* (often also called Inside). An example segmentation of the Finnish word *autoilta* (*from cars*) (*auto+i+lta*) is given by:

a	u	t	o	i	l	t	a
↓	↓	↓	↓	↓	↓	↓	↓
B	M	M	M	B	B	M	M

One can define more fine-grained labels, for example assigning a special label for the second position in a segment B^2, or a special label S for segments of length 1. A more detailed label set captures increasingly detailed structure at the cost of potentially overfitting the model to the training data, as statistics become sparser with larger label sets. Consequently, the optimal label set is task specific, depending on which positions in the segmentation differ in a predictable fashion, and data-set specific, as small data sets can be insufficient to learn detailed statistics.

4.2.1 Inference

The conditional random field defines a probability distribution for the label sequence conditioned on the input sequence. The most probable sequence can be calculated as:

$$\hat{\mathbf{y}}^* = \arg\max_{\mathbf{y}'} p(\mathbf{y}'|\mathbf{x}^*, \boldsymbol{\theta}) = \arg\max_{\mathbf{y}'} \frac{1}{Z(\mathbf{x}^*)} \prod_{t=2}^{|\mathbf{x}|} \exp \sum_{k=1}^{K} \left({\boldsymbol{\theta}_k}^\top f_k(y'_{t-1}, y'_t, \mathbf{x}^*, t) \right), \tag{4.17}$$

where we can once again ignore the normalizing constant $Z(\mathbf{x}^*)$ as it does not affect maximization. It can be seen that regarding the y variables this expression is structured in the same way as Expression (3.38) for hidden Markov models. Consequently, this expression can also be calculated efficiently with the Viterbi algorithm, utilizing the following iteration:

$$\delta_t(j) = \max_{i \in \mathcal{Y}} \exp \sum_{k=1}^{K} \left({\boldsymbol{\theta}_k}^\top f_k(i, j, \mathbf{x}^*, t) \right) \delta_{t-1}(i), \tag{4.18}$$

where $\delta_0(i) = 1$. The backward iteration is then equivalent to the one for HMMs, presented in Section 3.3.1.

4.2.2 Parameter Estimation

The optimal parameters for linear-chain conditional random fields cannot be calculated in closed form for a given data set $\mathcal{D} = \{x^{(i)}, y^{(i)}\}_{i=1}^{N}$, in contrast to the HMM. Instead, numerical optimization methods are utilized. Parameter estimation can be performed by maximizing the conditional likelihood or alternatively by classification-based methods, such as the averaged perceptron algorithm [Collins, 2002].

Maximum Likelihood The function $l(\boldsymbol{\theta})$ to be optimized is the conditional log likelihood combined with a regularization term. This can be seen as performing Maximum a Posteriori inference with a prior $p(\boldsymbol{\theta})$:

$$l(\boldsymbol{\theta}) = \sum_{i=1}^{N} \log p(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}, \boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) \tag{4.19}$$

$$= \sum_{i=1}^{N} \sum_{t=1}^{|\mathbf{x}^{(i)}|} \sum_{k=1}^{K} \left(\boldsymbol{\theta}_k{}^{\top} f_k(y_{t-1}^{(i)}, y_t^{(i)}, \mathbf{x}^{(i)}, t) \right) - \sum_{i=1}^{N} \log Z(\mathbf{x}^{(i)}) - \log p(\boldsymbol{\theta}) \tag{4.20}$$

A particularly attractive property of the CRF model is that the objective function is **convex**, and therefore every local optimum is also a global optimum (see e.g. Sutton and McCallum [2006]). Gradient-based methods are often employed, including stochastic gradient descent. Assuming L2-regularization with a Gaussian prior and a free parameter σ^2 to control the regularization, the gradient is given by:

$$\begin{aligned} \frac{\delta l}{\delta \boldsymbol{\theta}_k} = & \sum_{i=1}^{N} \sum_{t=1}^{|\mathbf{x}^{(i)}|} \left(f(y_{t-1}^{(i)}, y_t^{(i)}, \mathbf{x}^{(i)}, t) \right) \\ & - \sum_{i=1}^{N} \sum_{t=1}^{|\mathbf{x}^{(i)}|} \sum_{y,y'} f_k(y, y', \mathbf{x}_t^{(i)}, t) p(y, y'|\mathbf{x}_t^{(i)}, \boldsymbol{\theta}) \\ & - \sum_{k=1}^{K} \frac{\boldsymbol{\theta}_k}{\sigma^2} \end{aligned} \tag{4.21}$$

The first and the third terms are easy to compute. To calculate the marginal $p(y, y'|\mathbf{x}_t^{(i)}, \boldsymbol{\theta})$ in the second term requires a summation over all sequences $\mathbf{y}$. This summation can, however, be performed with the forward-backward iteration explained for hidden Markov models in Section 3.3.1. For linear-chain CRFs the forward and backward iterations

become

$$\alpha_t(j) = \sum_{l \in \mathcal{Y}} f(l, j, \mathbf{x}_t^{(i)}, t)\alpha_{t-1}(l) \tag{4.22}$$

$$\beta_t(j) = \sum_{l \in \mathcal{Y}} f(j, l, \mathbf{x}_t^{(i)}, t)\beta_{t+1}(l) \tag{4.23}$$

The normalizing constant $Z(x)$, required to compute the likelihood is given by $\sum_{l \in \mathcal{Y}} \alpha_{|\mathbf{x}^{(i)}|}(l)$ or $\beta_0(l)$. The marginal required for the gradient can then be calculated from the forward and backward variables:

$$p(y_t, y_{t-1}|\mathbf{x_t}, \boldsymbol{\theta}) \propto \alpha_{t-1}(y_{t-1}) f(y_t, y_{t-1}, \mathbf{x}_t^{(i)}, t)\beta_t(y_t) \tag{4.24}$$

Several gradient methods have been proposed for CRFs, including stochastic gradient descent and batch gradient. It is also popular to improve convergence speed by utilizing Newton's method that also takes into account the Hessian, that is, the matrix of second-order derivatives. Since the size of the Hessian is quadratic in the number of parameters in $\boldsymbol{\theta}$ it, however, becomes infeasible to calculate it for a larger number of parameters. Since CRFs often utilize many features this is a problem in practice. The problem can be resolved by employing approximate second order methods, such as limited-memory BFGS [Byrd et al., 1994].

Averaged Perceptron The averaged perceptron method for CRFs is an adaptation of the averaged perceptron algorithm in classification [Rosenblatt, 1958, Freund and Schapire, 1999]. The averaged perceptron calculates the most probable analysis $\hat{\mathbf{z}}^{(i)}$ for the ith training sample $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$ in the training data with the Viterbi algorithm. In case the model produces an error, that is, when $\hat{\mathbf{z}}^{(i)}$ does not match the training data analysis $\mathbf{y}^{(i)}$, the parameters are updated:

$$\theta_s = \theta_s + \sum_{t=1}^{|\mathbf{x}^{(i)}|} f_s(y_{t+1}^{(i)}, y_t^{(i)}, \mathbf{x}_t^{(i)}, t) - \sum_{t=1}^{|\mathbf{x}^{(i)}|} f_s(\hat{z}_{t+1}^{(i)}, \hat{z}_t^{(i)}, \mathbf{x}_t^{(i)}, t), \tag{4.25}$$

where θ_s is a single parameter in the parameter vector $\boldsymbol{\theta}$, and f_s its corresponding feature function. This algorithm can be shown to converge to zero training errors when a parameter vector allowing this exists. The theoretical results for generalization error can be improved by averaging θ_s between iterations [Freund and Schapire, 1999, Collins, 2002]. Consequently, the resulting method is known as the averaged perceptron.

Benefits of the averaged perceptron compared to gradient-based methods to calculate maximum likelihood is that it only requires Viterbi iteration over the data. This results in a fast algorithm that is easy to

implement. Moreover, it does not employ any hyperparameters, except the number of passes over the training data.

5. Unsupervised Learning of Allomorphy

Morphological segmentation is well-suited for modeling agglutinative structure in word formation, where forms are constructed by concatenating morphs. Examples of concatenative structure in English include *learn–learning* where the present participle formed by adding an *ing*-suffix, as well as *strong–stronger* where the comparative form is constructed by adding an *er*-suffix. However, even languages that are mainly characterized as agglutinative, often also contain fusional characteristics where morphemes undergo non-concatenative changes in particular contexts. Allomorphy refers to cases where an underlying morpheme-level unit has two or more morph-level surface realizations. Consider these other comparative forms which contain non-concatenative structure, *pretty–prettier*, *white–whiter*, and *hot–hotter*. Such fusional phenomena are problematic for morphological segmentation. In particular, the segmentation is incapable of preserving all relevant information about the word. In the segmentation *hot+ter* the connection to the lemma *hot* is preserved while the connection to the affix *er* is lost, and vice versa for the segmentation *hott+er*. In principle, one could preserve both with the segmentation *hot+t+er*, but this is unintuitive as it seems natural to think that the word has two morphemes: one for the lemma *hot* and one for the comparative form. Moreover, it is difficult to see what independent meaning the additional segment *t* would convey. In the presence of deletions, segmentation becomes even less appealing. For instance, should the word *whiter* be segmented as *white+r* or *whit+er*? Neither alternative preserves both the stem and the suffix.

Generally, allomorphic variation ranges from minor changes to the allomorphs, as in the above examples, to more severely non-concatenative phenomena. For example, consider the inflection of strong verbs such as *bring–brought*, or *take-took* for which no sensible segmentation can main-

tain the connection between the instances of the stem. In extreme cases, different word-forms of the same lexeme are entirely dissimilar, such as *go* – *went*. Such suppletive cases are impossible to infer from orthographic properties. In contrast, at the other end of the spectrum, there is allomorphic variation that despite being non-concatenative is, nevertheless, somewhat regular. For example, the alternation between *y* and *i* is found widely. Consider that *fly–flier* and *pretty–prettier* both employ it, despite being different parts-of-speech. Such regularity is not unusual, perhaps because the allomorphic variation is often influenced by factors that are not morphological but, nevertheless, systematic, such as phonological or orthographic regularities.

In this chapter, we discuss Allomorfessor, our novel extension that adds non-concatenative modeling capabilities to Morfessor Baseline [Creutz and Lagus, 2002, Creutz et al., 2007]. Allomorfessor was originally proposed in Publication I and Publication II. Furthermore, we review literature related to the learning of non-concatenative morphological structure. In particular, we focus on unsupervised learning in the presence of allomorphy, although some of the techniques may be applicable to the modeling of other non-concatenative structure. Allomorfessor extends Morfessor Baseline with string transformations to model stem allomorphy in addition to concatenative structure.[1]

This chapter is structured as follows. In Section 5.1 we define the learning task. Then in Section 5.2 we review the literature on morphology learning in the presence of allomorphy. In Section 5.3 we review the Allomorfessor extension following Publication I and Publication II. In Section 5.4 we review the empirical results of Allomorfessor. Finally, in Section 5.5 we discuss the implications of the work and possible future directions.

5.1 Learning Task

In Section 2.2.1 we defined morphological analysis as the task of mapping a surface word to a sequence of morphological tags, either morphemes or tags based on morphosyntactic categories. This entails lemmatization and the identification of affixes with the same grammatical function. Here, we consider the role of allomorphy in the task of morphological analysis. Because an unsupervised method does not know the set of morphological

[1]Publication I and Publication II use the term *mutation* for string transformations. We change the terminology here in the interest of accuracy.

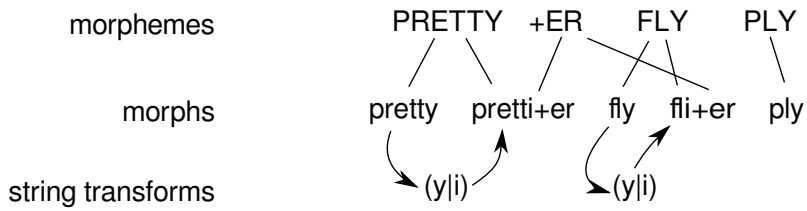

Figure 5.1. Illustration of non-concatenative morphological analysis with string transformations utilized to join allomorphs together. The string transformations follow the format presented in Section 5.3.1.

tags, that is lemmas and affix tags, it cannot return these directly. Instead the task becomes to identify latent morphs, the surface forms of morphemes, and grouping together all morphs that are allomorphs of the same morpheme. For example, we need to discover that the morph *fli* in the segmented word *fli+er* refers to the same morpheme as the single morph in the word *fly*. In contrast to the segmentation part that has been studied widely, the grouping of allomorphs has received less attention. Consequently, in this chapter, we will mainly focus on this latter part.

It can be noted that there is a considerable overlap with this learning problem and the task of stemming which does not identify the true lemma of a word but returns a stem that is shared among word-forms of the lexeme.

The central task of learning allomorphy is to identify a *similarity* or *pairwise relation* that holds for all allomorphs of the same morpheme, but not for other morphs. In other words, for stems, rather than finding the true lemma of a derived form we need to identify features that relate the word-forms of the same lexeme. In principle, the relation can be identified by several methods. We will employ string transformations $t(\mathbf{x})$ which relate forms by transforming them into one another. In this framework, two words $\mathbf{x}_1$ and $\mathbf{x}_2$ can be related with a transformation for which $\mathbf{x}_2 = t(\mathbf{x}_1)$. Unless the class of transformations is constrained in some fashion, however, a transformations exist between any pair of strings $(\mathbf{x}_1, \mathbf{x}_2)$, and therefore the selection of an appropriate class of transformations is a central modeling decision.

The task, as illustrated in Figure 5.1, is then to find a graph of units, such that the units are morphs and they are related by transformations if and only if they are allomorphs of the same morpheme. Consequently, the abstract morphemes can be recovered by replacing all morphs in the graph with a single, shared tag.

This approach addresses some of the most frequent failings of morphological segmentation. However, it does not address all aspects of morphological analysis. What is left out of scope is syncretism, where the same surface morph can be produced by more than one morpheme, as well as the more exotic situation where a single morph is produced by more than one abstract morpheme. For example, the plural form *men* is analyzed as man_N $+PL$ despite not containing more than one morph. This latter case, however, is quite rare in the languages we will study, and it is not merely a problem for the presented string transformation model, but also for morphemic modeling in general.

5.2 Literature Review

In this section we review literature related to the unsupervised learning of morphology in the presence of allomorphy. As discussed in Section 2.1, allomorphy entails that the same abstract morpheme is realized as several distinct surface morphs. Meanwhile, a segmentation model is based on the simplifying assumption that a surface morph maps one-to-one to abstract units. Although it is well known that this is merely an approximation, it can be a reasonable starting point, especially if the language in question mostly employs concatenative structure [Goldsmith, 2001]. When moving beyond segmentation it becomes necessary to address the mapping between abstract and surface units in some way.

It turns out most work on this problem can be seen operating in a framework where there are two separate sub-problems: unit-identification and relating the units. We will review these approaches in Section 5.2.1. Then, in Section 5.2.2 we review alternative approaches. Finally, in Section 5.2.4 we present a concluding discussion of the the presented literature.

5.2.1 The Two-Step Approach

Most work in the literature on unsupervised learning of morphology in the presence of allomorphy follows the following schema:

1. Identify morphological units, e.g. morphs, stems, or base forms

2. Identify morphologically related units

We can see that step 1 can be implemented with similar or even identical techniques as morphological segmentation and, therefore, its algorithmic characteristics are familiar. In contrast, step 2 requires the association between units. Naively implemented, one would then consider the relation of every segment with every other segment. If V is the number of distinct units, then there are $V(V-1)/2$ pairs of these units and, consequently, naive processing of them requires computation time $O(V^2)$. Moreover, for a single unit only a handful of the other units are truly related. Although the truly related ones are usually orthographically similar, the converse is not always true, and unrelated forms can also display orthographic similarity. For example, consider the forms *pretty*, *prettier* and *pretend*, where the third form is quite similar orthographically despite not being related. Therefore, a central problem is deciding whether two orthographically similar forms are truly morphologically related, or whether the similarity is merely spurious.

The early work on learning allomorphy [Yarowsky and Wicentowski, 2000, Schone and Jurafsky, 2000, 2001, Baroni et al., 2002] focuses on the problem of identifying morphologically related word pairs. To this end, they utilize word features including orthographic similarity, word contexts, and word frequencies. Such an approach can be seen as bypassing step 1, operating directly on word forms, and focusing on step 2. Although, for example, Yarowsky and Wicentowski [2000] report impressive accuracies for English, applying these methods to languages with richer morphologies is nontrivial.[2] The applicability of these methods depends crucially on the assumption that the words under study are frequent enough to enable the calculation of the employed statistics. For morphologically rich languages, however, the central problem for statistical modeling in general is precisely that one cannot observe all forms even in a very large corpus. Furthermore, they assume a single relatedness when a complex compound may very well have several, one for each morph in the compound.

Later work also includes step 1 and instead of operating directly on the input words they, rather, try to relate latent morphs or stems [Goldwater and Johnson, 2004, Goldsmith, 2006, Demberg, 2007, Dasgupta and Ng, 2007, Naradowsky and Goldwater, 2009, Lignos et al., 2010, Lignos, 2010]. We follow this line of work in Publication I and Publication II. Operating

[2]It should also be noted that the approach by Yarowsky and Wicentowski [2000] is not unsupervised, but employs several dictionary-based sources of supervision

on the morph level, however, makes the problem much more challenging. Not only must we identify relations between observed words, but between *latent* morphs. In other words, we must both segment each word form correctly and then for the *correct* morph identify its related morphs. A limitation in most of the suggested methods for this task is that they do not model general concatenative morphology, including, e.g., compound words. Goldwater and Johnson [2004], Goldsmith [2006], Naradowsky and Goldwater [2009] focus on stem–suffix morphology. Some approaches move beyond stem–suffix morphology, but are still more constrained than full concatenative morphology. Bernhard [2009], Lignos et al. [2010] allow multiple suffixes but only one stem, whereas Dasgupta and Ng [2007] allow also several stems, but not affixes between stems.

When working with morphologically rich languages and full concatenative morphology, the sparse statistics are a key part of the problem. As an illustration we can take the work in Lignos [2010] that extends [Lignos et al., 2010] to allow for missing intermediate forms of a compound. The results are mixed, giving increased scores for Finnish and Turkish, but reduced scores for English and German.

5.2.2 Alternative Approaches

Before we review in more detail the work closest to ours we will discuss some alternative approaches. Some authors have suggested that only step 2 is important and one should forgo abstract morphemes and only model the morphological relations between words [Neuvel and Fulop, 2002]. A benefit of this view is conceptual elegance. Allomorphy is not an additional phenomenon to model, but both concatenative and non-concatenative structure is modeled by transformations. However, we note that the abstract morphemes can also be interpreted as defining a relation graph with the words as nodes and edges defined by shared morphemes. Furthermore, learning the concatenative structure has received more attention and is therefore more mature. An interesting intermediate case is the method presented by Bernhard [2009] which takes the observed relations between words as the starting point but in the end learns a representation that can be output as abstract units. In contrast to the two-step approach where morphs and their relations are latent, Bernhard [2009] considers the relations between word strings as observed data, allowing all transformations except the very least frequent ones. The problem of discovering latent morphemes is then performed by clustering the graph

defined by the relations, assuming that word-forms of lexemes should form densely connected components.

Another alternative approach is to not depend on exact string matches when identifying units in step 1. Many allomorphs are similar if not identical, such as *hot* and *hott*. As a further example, consider the Finnish inessive suffix *ssA*, where the *A* is realized by different vowels to maintain vowel harmony within the word, for instance *auto+ssa* (in car) and *kylä+ssä* (in village). The suffix could elegantly be modeled by a unit that matches two *s*-characters followed by either *a* or *ä*. Such an approach is taken by Demberg [2007] who explicitly learns character classes to model umlauts. A different approach to learning non-exact orthographic features is employed by De Pauw and Wagacha [2007] who apply a feature-based classifier to learn discriminative word features. The classifier is trained in such a way that the target class is the word identity and the input features are word substring-features. The output probabilities of the classifier for a given input word can then be interpreted as expressing word similarity. They report results on a small data set where the method manages to find meaningful connections between words that are related but quite distant orthographically. Although learning word features based on an auxiliary prediction task has recently been successful in several other natural language processing tasks [Turian et al., 2010, Collobert et al., 2011, Al-Rfou et al., 2013], in this context the approach has, however, not been developed further.

Recently, Botha and Blunsom [2013] proposed a model based on Adaptor Grammars for non-concatenative morphology. The modeled structure is, however, related to templatic morphology as found in, for example Arabic, rather than allomorphic variation.

A recent related line of work is the supervised and semi-supervised learning full morphological paradigms from online dictionaries and predicting all the morphological form of unseen lexemes [Dreyer and Eisner, 2011, Durrett and DeNero, 2013, Ahlberg et al., 2014]. The key difference to our work is that here it is assumed that the morphosyntactic categories, such as 1st person plural, are known a priori, and that all forms of a particular word are observed at training time. What is, however, similar with unsupervised learning of allomorphy is the modeling of related forms by utilizing string transformations [Dreyer and Eisner, 2011]. Interesting departures from string transformations that could be applied to unsupervised learning of allomorphy as well includes the application of

log-linear modeling for the dependency between word forms [Dreyer and Eisner, 2011] as well as modeling all forms using a generic schema extracted from the data [Ahlberg et al., 2014]. In the work by Dreyer and Eisner [2011], rather than relating words by explicit transformations, a model of the joint distribution between words is constructed. The joint distributions is defined over string pairs, where the strings are of two particular forms of the same verb. The model is based on a log-linear model that allows the utilization of rich features that operate on the two aligned strings. The joint distribution is then formed by marginalizing over alignments.

5.2.3 Differences in Methods Identifying Latent Relations Between Latent Units

We will now review in more detail the work employing the two steps introduced in Section 5.2.1. The methods can be categorized based on the following distinctions.

Procedural or Model-based Most methods for morphological learning of allomorphy are *procedural* rather than employing an explicit model or objective function. Procedural methods include [Demberg, 2007, Dasgupta and Ng, 2007, Lignos et al., 2010, Lignos, 2010]. Explicitly model-based approaches include our work in Publication I and Publication II as well as [Goldwater and Johnson, 2004, Goldsmith, 2006, Naradowsky and Goldwater, 2009]. Bernhard [2009] employs an explicit objective function for the intermediate graph clustering, but not when deriving the final output representation.

Extension of Segmentation or Purely Transformation-Based Most work on unsupervised learning of morphology in the presence of allomorphy is based on extending an existing method for unsupervised learning of morphological segmentation. Goldwater and Johnson [2004] and Goldsmith [2006] extend Linguistica [Goldsmith, 2001], Demberg [2007], Dasgupta and Ng [2007] extend the segmentation model by Keshava and Pitler [2006], and Naradowsky and Goldwater [2009] extend the segmentation model of Goldwater et al. [2006]. Similarly, our work in Publication I and Publication II extends Morfessor [Creutz and Lagus, 2002, Creutz et al., 2007]. In contrast, Bernhard [2009], Lignos et al. [2010], Lignos [2010] are entirely based on string transformations and model concatenative structure merely as a special case.

String Transformations Class Levenshtein (edit) distance is employed to calculate the information required to derive the string transformations by Publication I, Publication II, Demberg [2007], and Bernhard [2009]. All mentioned authors then develop their own format for the transformation. In addition, Demberg [2007] learns character equivalence classes to model umlauting. In our work, string transformations are constrained to disallow insertions, such that the string transformations are only applied for non-concatenative structure and not, for instance, suffixing. Demberg [2007] employs similar operations, but allows insertions as well. As Bernhard [2009] merely employs the string transformations as evidence of connectedness, the transformations allow for very general changes. Goldwater and Johnson [2004], Naradowsky and Goldwater [2009] employ string transformations that occur at the morpheme boundaries (stem–suffix). The transformations consist of a single insertion, substitution or deletion operation. A key difference to other approaches suggested in the literature, Goldwater and Johnson [2004], Naradowsky and Goldwater [2009] employ context descriptors that encode when a particular transformation is applicable. The contexts consist of the characters surrounding the morph boundary, following the ideas developed by Chomsky and Halle [1968] that have been widely applied in rule-based morphological analysis (see e.g. Karttunen and Beesley [2005]). In contrast to the approaches based on the Levenshtein edit operations, Goldsmith [2006] considers only single character deletions. Similarly, Dasgupta and Ng [2007] considers single substring differences. Lignos et al. [2010], Lignos [2010] employs a straightforward scheme where the string transformation consists of two character lists: the substring deleted from and the substring added to the base form, respectively. For example, *make–making* becomes (*e*, *ing*). The benefit of this approach is that concatenative and non-concatenative structure alike can be modeled.

Distinguishing true Morphological Relations from Spurious Relations The early work [Yarowsky and Wicentowski, 2000, Schone and Jurafsky, 2000, 2001, Baroni et al., 2002] employ a combination of word context, frequency and orthographic features. However, the later work [Goldwater and Johnson, 2004, Goldsmith, 2006, Demberg, 2007, Dasgupta and Ng, 2007, Naradowsky and Goldwater, 2009, Lignos et al., 2010, Lignos, 2010] and Publication I and Publication II only employ orthography through string transformations and their frequency. Perhaps, this is motivated by sparse statistics, but potentially word context features could, neverthe-

less, be of utility. Especially, as such features have been found beneficial to the accuracy of unsupervised morphological segmentation [Lee et al., 2011].

As most methods are procedural, they employ specific heuristics. In contrast, our work in Publication I and Publication II as well as [Goldwater and Johnson, 2004, Goldsmith, 2006] are based on the minimum description length-principle [Rissanen, 1989]: if a string transformation enables compact storage of the observed forms, then it is applied. This is primarily related to the frequency of the string transformation, but other considerations, such as string lengths of the transformation and the morphs that can be left out of the lexicon affect the result as well. Naradowsky and Goldwater [2009] similarly employ model-based criteria. In contrast to previous work, they employ a prior that is not derived from minimum description length.

The procedural work employs several different heuristics. Demberg [2007] searches for stem candidates as strings which combine with frequent suffixes. Allomorphic variation is identified by examining suffix groups that share prefixes. Edit-distance is calculated on the groups. Real transformation rules are then identified by frequency of the rule. Dasgupta and Ng [2007] identify candidate allomorphs by identifying words that vary only by a single substring and a suffix. A rule is then induced to change the varying substring. To filter out bad candidates, frequency is employed, as well as a specificity measure for the rule which favors rule sets that consistently substitute one character for another, rather than having several alternative rules for the same character. Lignos et al. [2010] discovers transformations based on the most frequent suffix substrings which are then filtered based on how many base–derived pairs can be constructed. To limit the search space, transformations are constrained to operate on maximally 5 character suffixes and transforming only between frequent ones.

5.2.4 Discussion

In the previous sections, we have discussed the literature on unsupervised learning of morphology in the presence of allomorphy. We identified several different approaches to the problem. In general, however, none of the presented methods have reached the popularity enjoyed by unsupervised segmentation methods [Harris, 1955, Goldsmith, 2001, Creutz and Lagus, 2007, Roark and Sproat, 2007, Hammarström and Borin, 2011], but

rather represent a small niche in the morphology learning literature. This is somewhat surprising, since several of the presented methods report improved empirical performance compared to unsupervised methods for morphological segmentation. Generally, many of the reviewed methods have not participated in systematic evaluations such as the Morpho Challenge competitions [Kurimo et al., 2010], and such comparisons would be needed to provide more insight on the relative performance difference compared to state-of-art unsupervised morphological segmentation methods. In Morpho Challenge 2010, the best performing unsupervised morphological analysis method was the method by Lignos [2010]. It demonstrated improved performance over state-of-art unsupervised morphological segmentation methods for English and Finnish, but not for German and Turkish.

5.3 Allomorfessor

In this section we present Allomorfessor, the unsupervised method for learning morphology presented first introduced in Publication I and later adapted in Publication II. The presentation will follow the version presented in Publication II, to which we will refer to as Allomorfessor, as the version presented in Publication I (Allomorfessor Alpha) is conceptually very similar, but does not perform nearly as well. Allomorfessor extends Morfessor Baseline [Creutz and Lagus, 2002, Creutz et al., 2007] from morphological segmentation to morphological analysis by adding string transformations to the generative model. Allomorfessor connects allomorphs by mapping morphs to one another with string transformations. In the learning problem under study, the morphs and the string transformations are both latent and must, consequently, be inferred from the data. Its objective function is similar to that of Morfessor Baseline and is based on the minimum description length principle [Rissanen, 1989]. The intuition behind the model formulation is that when introducing string transformations into a minimum description length model, it will favor true allomorphic relations over spurious ones, because a compactness criterion will prefer transformations that occur systematically. For example, when storing the morphs *pretty, pretti, happy, happi* a transformation that substitutes *y* with *i* can improve compactness by eliminating the need for storing two of the above four forms. Allomorfessor does not employ any other form of information source to guide its learning except this

orthography-based compression objective.

In addition, Allomorfessor has two design goals: First, it should be applicable to large word lists. Second, it should generalize to infrequent forms, as detailed models of words are most useful for infrequent forms, whereas downstream models can infer the properties of frequent words directly in a task-specific fashion. The choices made in Allomorfessor reflect these goals. The first design goal indirectly implies that quadratic algorithms, such as naively comparing all pairs of words are out of the question. The second design goal precludes depending on statistics that are only available for frequent words, such as word contexts, instead concentrating on string transformations that can be inferred for frequent and infrequent forms alike, and even applied to unseen words.

Next, we present how words are produced from a lexicon of morphs and string transformations. Then, we present how this process is modeled probabilistically as well as how model parameters are estimated during training, and how the model is applied to the new words. Finally we review some experiments and discuss the implications of the work.

5.3.1 Generating Words with String Transformations

Allomorfessor extends the generative process of Morfessor Baseline [Creutz and Lagus, 2002, Creutz et al., 2007]. As presented in Section 4.1.1, Morfessor Baseline generates a word $\mathbf{x}$ by concatenating an arbitrary number of morphs $\mathbf{z}_j$, following Expression 4.5. In addition, to model non-concatenative structure, the Allomorfessor model introduces string transformations $\mathbf{t}$ that operate on the morphs. Consequently, the analysis $\mathbf{z}$ of the word $\mathbf{x}$ consists of an arbitrary number of pairs of morphs $\mathbf{m}$ and string transformations $\mathbf{t}$ such that $\mathbf{z}_j = (\mathbf{m}_j, \mathbf{t}_j)$. The model considers two morphs to be allomorphs of the same morpheme if one is generated as a transformation of the other. The analysis is visualized by placing a string transformation between each morph, where the transformation operates on the morph to its left: $\mathbf{m}_1 + \mathbf{t}_1 + \mathbf{m}_2 + \mathbf{t}_2 \ldots \mathbf{m}_{n-1} + \mathbf{t}_{n-1} + \mathbf{m}_n$. The word $\mathbf{x}$ is then produced as:

$$\mathbf{x} = \mathbf{t}_1(\mathbf{m}_1) \circ \mathbf{t}_2(\mathbf{m}_2) \circ \cdots \circ \mathbf{t}_{n-1}(\mathbf{m}_{n-1}) \circ \mathbf{m}_n \tag{5.1}$$

In other words, all morphs except the final one will be affected by some transformation. The final morph is exempt because the focus is on stem allomorphy rather than suffix allomorphy. Concatenative structure is modeled with the empty transformation $\mathbf{t}^{id}$ which is an identity mapping

$\mathbf{t}^{id}(\mathbf{m}) = \mathbf{m}$.

We will now discuss the choice of the class of string transformations. The following aspects need consideration:

1. As a consequence of how we have defined the generation of words, it is necessary for the string transformation to be *executable*. That is, we should be able to take any morph $\mathbf{m}$ and apply any transformation $\mathbf{t}$ to it with a well-defined result.

2. The string transformation should not interfere with the generation of concatenative structure. For example, suffixation can in principle be modeled both as a concatenation *fast* $\circ$ *er* or via a transformation $\mathbf{t}^{+er}$(*fast*), where $\mathbf{t}^{+er}$ refers to a hypothetical transformation that adds *er* at the end of the input morph.

3. Because of the minimum description length objective function, it is important that what we intuitively consider the same transformation is also realized in practice with the same string transformation for different stems. If this is not true, compactness is not achieved and the model will not prefer such transformations. This implies that trivial aspects, such as word length, should not affect the transformation.

4. The string transformations that occur in true allomorphic variation (*pretty – prettier*) should generally tend to be shorter in code length than string transformations corresponding to spurious relations (*ply – fly*), or at least the spurious transformations should tend to be infrequent.

These aspects have been approached in Allomorfessor by defining a custom executable string transformation class. Similarly to [Yarowsky and Wicentowski, 2000, Goldwater and Johnson, 2004, Demberg, 2007, Bernhard, 2009, Naradowsky and Goldwater, 2009] the string transformations are based on the edit operations employed in Levenshtein (edit) distance, insertion, substitution, and deletion (see section 3.5). We utilize the property of Levenshtein distance that its calculation, as a side product, efficiently computes the minimal edit operations required to transform one string into another. Of these edit-operations we employ substitution and deletion. We do not allow insertion in order to ensure that the string transformations cannot be used to model concatenative structure. This

implies that not all morphs can be transformed into one another. Rather than disallowing insertions, Chan [2008], Lignos et al. [2010], Lignos [2010] model all morphological operations, including concatenative structure, with string transformations. This approach is conceptually elegant; however, unlike allomorphic variation, learning of the concatenative structure has been studied extensively within unsupervised learning of morphological segmentation. To leverage this previous work we, therefore, attempt to extend a segmentation model, rather than redefining the task to one of pure string transformations [Chan, 2008, Lignos et al., 2010, Lignos, 2010], or a purely relational one [Neuvel and Fulop, 2002].

Table 5.1. The string transformation operations and some examples in Finnish. Note that the operations are applied from right to left.

Operation	*Notation*	*Description*
substitution	k`x|y`	Substitute k:th character `x` with `y`
deletion	-k`x`	Remove k:th `x` character

(k is omitted when $k = 1$)

Source	*String transformation*	*Target*
`kenkä` (shoe)	`(k|g)`	`kengä` (e.g. `kengä+ssä`, in a shoe)
`tanko` (pole)	`(k|g)`	`tango` (e.g. `tango+t`, poles)
`ranta` (shore)	`(-a t|n)`	`rann` (e.g. `rann+oi+lla`, on shores)
`ihminen` (human)	`(2n|s)`	`ihmisen` (human's)

The string transformations employed in Allomorfessor are defined as follows. The transformations `t` consist of a sequence of position independent substitution and deletion operations. Position independence is desirable as we want to apply the same string transformation to morphs of different lengths (*fly–flier, amplify–amplifier*). However, the edit-operations resulting from the calculation of Levenshtein-distance are position dependent. We calculate position independent string transformations as follows: First, we note that in the studied languages, allomorphic variation is more common towards the end of a morph. Consequently, the string transformation proceeds from right to left. Second, to achieve position independence, the target position of the operation is identified by specifying which character the operation should target. Since several instances of the same character may occur, we additionally specify which one of the repeating characters to target. The notation employed for the string

transformation operations is shown in Table 5.1 together with some examples. Several operations can be combined in sequence, in which case the target character matching proceeds from the position where the previous operation ended. The analysis of a whole word is then written by placing the transformation between each morph. For example, *prettier* would be correctly analyzed as *pretty + (y | i) + er*. For concatenative structure, the empty transformation $\mathbf{t}^{id}$ is denoted with empty parentheses: For instance, *fast + () + er*.

5.3.2 Generative Model

As described in Section 4.1.1, the Morfessor Baseline model generates each word by concatenating morphs, that is substrings, that are drawn independently from the morph lexicon. In Allomorfessor this process is extended in such a way that in addition to morphs, the lexicon also contains string transformations. There is an empty transformation for concatenative structure. The model constructs the word $\mathbf{x}^{(i)}$ by independently generating morphs $\mathbf{m}$, and then, conditioned on the previously generated morph, samples a string transformation $\mathbf{t}$. The string transformation operates on the morph on its left. In contrast, the probabilistic model generates the string transformation conditioned on the morph to the right of the string transformation. The generative process proceeds over the word $\mathbf{x}^{(i)}$ in reverse. It begins by generating the final, rightmost, morph in the word and then proceeds from the right to the left. The model generates a morph $\mathbf{m}_j$ and then conditioned on $\mathbf{m}_j$ it generates a string transformation $\mathbf{t}_j$. It should be noted that $\mathbf{t}_j$ operates not on $\mathbf{m}_j$, but rather on the previous morph $\mathbf{m}_{j-1}$ generated in the next generation step. Conditioning on the right morph, rather than the left one, is based on the intuition that suffix morphs tend to be more frequent, leading to less sparse statistics for estimating the conditional distribution. In Allomorfessor we extend the analyses $\mathbf{z}_j$ such that it contains pairs of morphs and string transformations. For example, if the word $\mathbf{x}^{(i)}$ is `prettier`, a possible analysis is $\mathbf{z}_j = [(\mathbf{t}^{id}, \texttt{pretty}), (\texttt{(y|i)}, \texttt{er})]$. The first string transformation has no morph on its left, and is therefore an empty transform by definition. This

results in the optimization problem:

$$\hat{\boldsymbol{\theta}}_{MAP} = \arg\max_{\boldsymbol{\theta}} p(\mathcal{U}|\boldsymbol{\theta})p(\boldsymbol{\theta}) \tag{5.2}$$

$$p(\mathcal{U}|\boldsymbol{\theta})p(\boldsymbol{\theta}) = p(\boldsymbol{\theta}) \prod_{i=1}^{N} \sum_{\mathbf{z}^{(i)} \in \text{SEG-TR}(\mathbf{x}^{(i)})} p(\mathbf{x}^{(i)}|\boldsymbol{\theta}, \mathbf{z}^{(i)}) p(\mathbf{z}^{(i)}) \tag{5.3}$$

$$p(\mathbf{x}^{(i)}|\boldsymbol{\theta}, \mathbf{z}^{(i)}) = \prod_{(\mathbf{t}_j, \mathbf{m}_j) \in \mathbf{z}^{(i)}} p(\mathbf{t}_j|\mathbf{m}_j, \boldsymbol{\theta}) p(\mathbf{m}_j|\boldsymbol{\theta}), \tag{5.4}$$

$$\tag{5.5}$$

where we can see that Expression (5.2) is equivalent to the one of Morfessor Baseline in Section 4.1.1. Expression (5.3) differs only in that the latent analysis $\mathbf{z}^{(i)}$ varies over the set SEG-TR($\mathbf{x}^{(i)}$) of all the combinations of morph strings and string transformations that produce the word $\mathbf{x}^{(i)}$, rather than merely the set of all segmentations SEG($\mathbf{x}^{(i)}$). In addition, the generation of words in Expression 5.4 is adapted to add string transformations.

The prior $p(\boldsymbol{\theta})$, described in detail in [Virpioja and Kohonen, 2009], is very similar to that of Morfessor Baseline. For string transformations, the prior follows an MDL-based derivation similar to that of morphs. Informally, the fewer different morph or string transformation types there are, the higher the prior probability is. Moreover, shorter morphs and string transformations have higher probabilities than longer ones.

We will now briefly discuss properties of the model. Generally, most aspects are straightforward extensions of Morfessor Baseline. However, the conditional generation of the string transformation $\mathbf{t}$ is less straightforward than independent generation. In fact, independent generation of transformations was attempted in early work [Kohonen et al., 2009]. It turned out to result in severe under-segmentation and, consequently, much worse performance than Morfessor Baseline. It can be noted that when employing the conditional distribution as in Allomorfessor, the model reduces exactly to Morfessor Baseline if all string transformations are clamped to the empty transformation. Therefore, it can be argued that the conditional generation of string transformations is an extension closer to the original Morfessor Baseline method.

5.3.3 Parameter Estimation

For learning the model parameters, we apply an iterative greedy algorithm similar to the one used by Morfessor Baseline which we reviewed in Section 4.1.1. The algorithm finds an approximate solution for the

MAP problem in Equation (5.2). The Allomorfessor parameter estimation method, shown in Algorithm 3, is equivalent to that of Morfessor Baseline with a few exceptions: First, the latent analysis $\mathbf{z}$ is extended to allow for string transformations. Second, the set of considered analyses $\mathcal{Z}_m$ for each morph $\mathbf{m}$ now, in addition to two-way splits, also contain analyses with string transformations. Third, the ML update of the parameters $\hat{\boldsymbol{\theta}}$ now includes also the string transformation parameters.

Algorithm 3 The learning algorithm

Initialize analysis $\hat{\mathbf{Z}} = \{\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \dots \mathbf{z}^{(N)}\}$ and model parameters $\hat{\boldsymbol{\theta}}$
while $\prod_{i=1}^{N} p(\mathbf{x}^{(i)}|\hat{\boldsymbol{\theta}}, \hat{\mathbf{z}}^{(i)})p(\hat{\boldsymbol{\theta}})$ increases sufficiently **do**
  for $\mathbf{x}^{(i)} \in \mathcal{U}$ in random order **do** optimize($\mathbf{x}^{(i)}$)
end while
function optimize($\mathbf{m}$)
  $\mathcal{Z}_m \leftarrow \left[\mathbf{m}\right] \cup \left[(\mathbf{m}_{<1..l>}, \mathbf{m}_{<(l+1)..|\mathbf{m}|>}) : l \in 1, ..., |\mathbf{m}| - 1\right]$
    $\cup$ transformed_analyses($\mathbf{m}$)
  for $k = 1 \dots K$
    let $\hat{\mathbf{z}}_{mk}$ be the kth element of $\mathcal{Z}_m$
    $\hat{\mathbf{Z}}_k \leftarrow$ update_analyses($\hat{\mathbf{Z}}, \mathbf{m} \leftarrow \hat{\mathbf{z}}_{mk}$)
    $\hat{\boldsymbol{\theta}}_{\hat{\mathbf{Z}}_{mk}} \leftarrow \arg\max_{\boldsymbol{\theta}} p(\mathcal{U}|\boldsymbol{\theta}, \hat{\mathbf{Z}}_{mk})$ (ML estimate for $\boldsymbol{\theta}$ given $\hat{\mathbf{Z}}_{mk}$)
  $k_{best} \leftarrow \arg\max_{k=1...|\mathcal{Z}_m|} \prod_{i=1} p(\mathbf{x}^{(i)}|\hat{\boldsymbol{\theta}}_{\hat{\mathbf{Z}}_{mk}}, \hat{\mathbf{z}}^{(i)}_{mk})p(\hat{\boldsymbol{\theta}}_{\hat{\mathbf{Z}}_{mk}})$
  $\hat{\mathbf{Z}} \leftarrow \hat{\mathbf{Z}}_{k_{best}}$
  $\hat{\boldsymbol{\theta}} \leftarrow \hat{\boldsymbol{\theta}}_{\hat{\mathbf{Z}}_{mk_{best}}}$
  if $\hat{\mathbf{z}}_{mk_{best}}$ involved a split at l **then** optimize($\mathbf{m}_{<1..l>}$); optimize($\mathbf{m}_{<(l+1)..|\mathbf{m}|>}$)

Algorithm 4 transformed_analyses($\mathbf{m}$)

Initialize analysis set $\mathcal{Z} =$
for $l \in 1, ..., |\mathbf{m}| - 1$ **do**
  if $|\mathbf{m}| >= 4 \wedge |\mathbf{m}_{<(l+1)..|\mathbf{m}|>}| <= 5 \wedge p(\mathbf{m}_{<(l+1)..|\mathbf{m}|>}|\boldsymbol{\theta}) > 0$ **then**
    if $|\mathbf{m}| > 6$ **then** *difflen* $\leftarrow 4$ **else** *difflen* $\leftarrow 3$
    baseforms $\leftarrow \{\mathbf{v}^{(k)} \in \mathcal{U} : \mathbf{v}^{(k)}_{<1..(|\mathbf{m}|-\textit{difflen})>} = \mathbf{m}_{<1..(|\mathbf{m}|-\textit{difflen})>}\}$
    Calculate transformation $\mathbf{t}^{(k)}$ from each $\mathbf{v}^{(k)} \in$ *baseforms* to $\mathbf{m}$
    $\mathcal{Z} \leftarrow \mathcal{Z} \cup \left[(\mathbf{v}^{(k)}, \mathbf{m}_{<(|\mathbf{v}^{(k)}|+1)..|\mathbf{m}|>}, \mathbf{t}^{(k)})\right]$
  end if
end for
return $\mathcal{Z}$ sorted by l and descending $|\mathbf{v}^{(k)}|$

Compared to Morfessor Baseline, the set of considered analyses $\mathcal{Z}_m$ for

each morph $\mathbf{m}$ is now potentially very large, as any morph of adequate length can be transformed into another one with a sufficiently complex string transformation. However, morphs related by complex string transformations are unlikely to correspond to true allomorphic relations. For computational reasons, the parameter estimation method cannot consider too large a set of analyses for each morph. Therefore, a set of heuristic restrictions are employed to limit the search, as shown in Algorithm 4. The restrictions are as follows: The morph and its potential base form have to begin with a shared substring, the base form has to occur as a word in the training set, and the suffix has to be present in the lexicon. Finally, only K analyses per morph are tested, which results in time complexity $O(K^2N)$ for one epoch of the.

5.3.4 Inference

The inference algorithm finds the best analysis $\mathbf{z}^* = \arg\max_{\mathbf{z}} p(\mathbf{z}|\mathbf{x}, \hat{\boldsymbol{\theta}})$ for new words using a similar algorithm as the extended Viterbi algorithm that Morfessor Baseline employs, described in Section 4.1.1. With string transformations, the inference algorithm requires further extension. As string transformations are conditioned on the suffixes, it is easier to run the algorithm from right to left. A two-dimensional grid is required as there can now be several morphemes that produce the same surface string. The rule for updating the grid value for s_i is

$$\alpha(s_i, \hat{\mathbf{m}}_{ij}) = \max_{j \in [i+1, |w|]} \Big\{ \max_{\mathbf{m} \in s_j} \Big\{ \max_{\mathbf{t} \in \Delta} \big\{ \alpha(s_j, \mathbf{m}) P(\mathbf{t}|\mathbf{m}, \boldsymbol{\theta}) P(\hat{\mathbf{m}}_{ij}|\boldsymbol{\theta}) \big\} \Big\} \Big\}, \quad (5.6)$$

where $\hat{\mathbf{m}}_{ij}$ is a morpheme that produces the letters between i and j when modified by the string transformation $\mathbf{t}$. Only those string transformations Δ that are observed before $\mathbf{m}$ need to be tested, otherwise $P(\mathbf{t}|\mathbf{m}, \boldsymbol{\theta}) = 0$. For the morphemes that are not observed before, we use an approximate cost of adding them into the lexicon. The worst case time complexity for the algorithm is $O(MD|w|^2)$. In practice, however, the number of morphemes M and string transformations D tested in each position is quite limited.

5.4 Experiments

The method was evaluated in Morpho Challenge 2009 [Kurimo et al., 2009b] which included three competitions: Competition 1, comparison to a linguistic gold standard on English, Finnish, German, Turkish and

Arabic data; Competitions 2 and 3, application evaluations in information retrieval and machine translation, respectively. The information retrieval data sets are in English, Finnish and German whereas the machine translation data is Finnish-English and German-English parallel text. All three evaluations measure the overall performance of the proposed analysis without directly measuring the effects of the string transformations. This is, however, compensated for by employing Morfessor Baseline as a reference method, as it is a very similar method except for the string transformations.

For Competitions 1 and 2 we trained the model with the Competition 1 data. We filtered out all words that occurred fewer than $T = 2$ times in the corpus to shorten training times and remove noise such as misspelled words, where T denotes the cutoff threshold below which frequency a word is excluded from the training set. The training algorithm converged in 5–8 epochs and the total training time varied from hours up to one week (Finnish). After training the model, we analyzed all the words with the Viterbi algorithm. For Competition 3, we used the provided data sets for training without any filtering and then applied the Viterbi algorithm.

We compare Allomorfessor as presented here, following Publication II, to reference methods, namely its earlier version from Publication I Allomorfessor Alpha as well as Morfessor Baseline. We train Morfessor Baseline both with and without the filtering of the words, such that only words occurring at least $T = 2$ are included.

In all models, the following priors and parameter settings were used: Morpheme length distribution was geometric with the parameter $p = n_W/(n_W + n_C)$, where n_W is the number of words and n_C is the number of characters in the training corpus. As the string transformation length prior we used a gamma distribution with both the scale and shape parameters set to one. The number of candidate analyses K considered for each morph during the training was 20.

In Competition 1, the algorithms were compared to a linguistic gold standard analysis and scored according to the Morpho Challenge metric (Section 2.2.2). Competition 2 compared the methods in an information retrieval system for English, Finnish and German and measure the Mean Average Precision of the resulting system. In Competition 3, the algorithms were applied in Finnish-to-English and German-to-English machine translation systems and BLEU scores [Papineni et al., 2002] were measured.

Table 5.2. Evaluation results for different versions of Allomorfessor and Morfessor. For Competition 1 results (C1), precision, recall and F-measure are given. Competition 2 (C2) is scored with mean average precision (MAP) and Competition 3 (C3) with BLEU-score. The row T denotes the cutoff threshold for how often words must occur in the training corpus in order to be included when training.

	Allomorfessor Alpha (-08)	Morfessor Baseline	Morfessor Baseline	Allomorfessor
T	1	1	2	2
English				
C1 precision	83.31%	74.93%	68.43%	68.98%
C1 recall	15.84%	49.81%	56.19%	56.82%
C1 F-measure	26.61%	59.84%	61.71%	**62.31%**
C2 MAP	-	38.61%	**38.73%**	38.52%
Finnish				
C1 precision	92.64%	89.41%	86.07%	86.51%
C1 recall	8.65%	15.73%	20.33%	19.96%
C1 F-measure	15.83%	26.75%	**32.88%**	32.44%
C2 MAP	-	44.25%	44.75%	**46.01%**
C3 BLEU	-	**28.61%**	-	28.56%
German				
C1 precision	87.82%	81.70%	76.47%	77.78%
C1 recall	8.54%	22.98%	30.49%	28.83%
C1 F-measure	15.57%	35.87%	**43.60%**	42.07%
C2 MAP	-	**46.56%**	**47.28%**	43.88%
C3 BLEU	-	31.19%	-	31.14%
Turkish				
C1 precision	93.16%	89.68%	85.43%	85.89%
C1 recall	9.56%	17.78%	20.03%	19.53%
C1 F-measure	17.35%	29.67%	**32.45%**	31.82%

5.4.1 Results

The results are shown in Table 5.2.[3] From the results of Competition 1, that is the linguistic evaluation, we can note the following: The current Allomorfessor version clearly outperforms the old one which tends to under-segment. It also outperforms Morfessor Baseline without the data filtering $T = 1$. When comparing to the Morfessor Baseline with the same data filtering applied, however, the results are inconclusive. Both Allomorfessor and Morfessor clearly benefit from the exclusion of rare words.

[3]We omit here the Arabic results, as the data sets seemed to be of questionable quality; no participating method outperformed the letters-baseline which splits each word into its constituent letters [Kurimo et al., 2009b].

With the filtering applied, Allomorfessor has higher precision and lower recall for all languages. For English, the increase in precision is sufficiently large to result in improved F-scores. In contrast, for the other languages, Allomorfessor has a lower F-score than Morfessor Baseline. All differences in F-scores for this method pair are statistically significant.

In Competitions 2 and 3 Allomorfessor and Morfessor had no statistically significant differences.

The amounts of string transformations employed by Allomorfessor are shown in Table 5.3. Generally, string transformations are used much more rarely compared to the amount of non-concatenative structure in the linguistic gold standard. The method, for example, stores the morph `pretti` instead of deriving it as `pretty (y|i)`. Some string transformations from the English and Finnish results are shown in Tables 5.4 and 5.5. To summarize, a large part of the string transformations correspond to a linguistic analysis. It is common, especially for Finnish, that a derived form functions as the base form. If the forms are derived from the same lexeme, however, this is not to be considered an error, since two related forms are joined together. Errors include not finding the linguistically correct suffix, using a more complex string transformation and suffix combination than necessary, and using a semantically unrelated base form. String transformations are also used commonly on misspelled words. Overall, the application of string transforms suffered mainly from low recall rather than low precision.

Table 5.3. The number of non-empty string transformations applied by Allomorfessor after the Viterbi analysis. String transformation usage is the number of non-empty string transformation tokens divided by the number of morph tokens.

Language	English	Finnish	German	Turkish
Transformation types	15	66	26	55
Transformation usage	0.18%	0.44%	0.17%	0.12%

5.5 Discussion

The presented extension from Morfessor Baseline to Allomorfessor was mostly straightforward. However, a key difference is the conditional rather than independent generation of the transformation given the morph on the right. Publication I introduced the method we have here referred to as Allomorfessor Alpha, in which the transformation was indeed generated

Table 5.4. String transformation types with example usage for English.

String transf.	*Count*	*Examples*	*Notes*
(-e)	1182	adhering: adhere (-e) ing	
(-y)	300	vulnerabilities: vulnerability (-y) ies	
		temporarily: temporary (-y) ily	
(-t)	120	affluence: affluent (-t) ce	
		bankrupcy: bankrupt (-t) cy	misspelled
(-a)	66	encyclopedic: encyclopedia (-a) c	
		hemophilic: hemophilia (-a) c	
(-i)	41	publshed: publish (-i) ed	misspelled
(-s)	35	euripidean: euripides (-s) a () n	
		diocletian: diocles (-s) tian	
(-o)	27	aspirating: aspiration (-o) g	suffix ing not found
(-n)	27	proletariat: proletarian (-n) t	
		restauration: restaurant (-n) ion	
(-c)	20	paraplegia: paraplegic (-c) a	
(t|c)	8	excellencies: excellent (t|c) ies	adj as base form
(a|s)	2	ljubljanska: ljubljana (a|s) ka	foreign
(-g)	1	licensintorg: licensing (-g) torg	oversegmented
(s|n)	1	sclerosing: sclerosis (s|n) g	suffix ing not found
(-h)	1	thorougbred: thorough (-h) bred	misspelled
(-a -y)	1	bulathkopitiya: bulathkopitya (-a -y) iya	foreign

independently. Generally, Allomorfessor Alpha under-segmented severely. The reason for this is that the independent generation makes substructure more expensive. In Morfessor Baseline, substructure implies that a single string is replaced by two shorter strings. However, in Allomorfessor Alpha, there are not two, but three units for each split; the string transformation must also be accounted for. It turns out that the encoding of the third unit increases the cost sufficiently to cause severe undersegmentation. The utilization of the conditional distribution resolves this issue, because for right hand morphs that only combine with the empty transformation, conditional generation yields no extra coding length cost. This can be seen for a right hand morph $\mathbf{m}_r$ that has only combined with the empty transformation $\mathbf{t}^{id}$, the ML parameters concentrate all probability mass to the empty transformation $p(\mathbf{t}^{id}|\mathbf{m}_r, \hat{\boldsymbol{\theta}}_{ML}) = 1$. Therefore, with conditional generation, the model reduces exactly to Morfessor Baseline if all string transformations are fixed to the empty transformation. We found, however, that in practice Allomorfessor yielded higher precision and lower recall than Morfessor Baseline. This could be caused by the extra cost of substructure from the conditional distributions in the

case where more than a single string transformation is generated in the context of a suffix.

This phenomenon raises the question of how well the MDL-prior is capable of determining the desired amount of segmentation automatically. Such questions will be discussed in Section 6.2.

Allomorfessor seems to perform very similarly to Morfessor Baseline when evaluating against linguistic analysis. Moreover, it was noted that the string transformations were not used nearly as often as in the gold standard analysis. The string transformations that the model did discover were, however, mostly regarded as correct ones. This would support the idea that the MDL-model can indeed select correct transformations over spurious ones, at least while the recall is low. However, when increasing recall, it may become more difficult to avoid spurious ones.

Finally, the question that should be asked is what the main purpose of the current task is. At the very least, one can think of 1) Improving application performance 2) Evaluating learning algorithms on a well understood task.

Regarding improving application performance, the achieved results are quite far from state-of-art rule-based methods. This is, to some extent, to be expected, as the learning algorithm is performing a quite difficult task that shares more with a linguist surveying a new language from texts alone, than with human language acquisition, since the latter happens in the presence of context that allows reasoning based on the meaning of words. Therefore, based on the current results, from a purely pragmatic perspective, learning morphology in an unsupervised fashion is not a good alternative to a good rule-based system. However, should the language under study lack such a system, or if such a system exists but is too expensive for the project in question, then unsupervised learning can provide an inexpensive alternative. In the context of Allomorfessor, the question remains whether one should apply segmentation or attempt to learn also non-concatenative structure. The current literature and the experiments presented in this dissertation cannot provide a conclusive answer, but further research is needed.

From an academic point of view, the current task provides a well-defined but exotic task for which to develop better machine learning methods. The introduction of relations between morphs, in the form of string transformations or otherwise, results in a much more challenging learning problem compared to the well-known segmentation problem. Furthermore,

the current state-of-art approaches are quite far from 100% accuracy according to the gold standard. This leaves an excellent potential for future progress.

Table 5.5. String transformation types with example usage for Finnish.

String transf.	*Count*	*Examples*	*Notes*
(-n)	7771	ahdingolla: ahdingon (-n) lla	
		aikojemme: aikojen (-n) mme	
(-i)	4096	anakronismeille:	
		anakronismi (-i) e () ille	(i|e) preferable
		desibelejä: desibeli (-i) ejä	(i|e) preferable
(-a)	2598	diakonissoja: diakonissa (-a) oja	(a|o) preferable
		eufemismi: eufemia (-a) smi	
(-t)	2507	fagotisti: fagotti (-t) sti	
		haltuunoton: haltuunotto (-t) n	
(-s)	1114	harvennuksen: harvennus (-s) ksen	
		yliherkkyydet: yliherkkyys (-s) det	
(-e)	939	vuosituhantista: vuosituhantiset (-e) a	
		viikattein: viikate (-e) tein	
(i|e)	675	videoprojektoreina:	
		video () projektori (i|e) ina	
		transistoreita: transistori (i|e) ita	
(-ä)	532	tulennielijöitä: tulennielijä (-ä) öitä	
		tulokertymien: tulokertymä (-ä) ien	
(a|i)	430	kaavailemia: kaavailema (a|i) a	
		juurevia: juureva (a|i) a	
(n|s)	428	hankkeeseesi: hankkeeseen (n|s) i	undersegmented
		diabeteksesi: diabeteksen (n|s) i	undersegmented
(a|e)	322	emigranttien: emigranttia (a|e) n	
		hajuharhojen: haju () harhoja (a|e) n	
(-k)	311	agnostikoksi: agnostikko (-k) ksi	
(-a -t)	232	murhissa: murhista (-a -t) sa	
(-n -i)	183	barrikadeja: barrikadin (-n -i) eja	
		kursseihen: kursseihin (-n -i) en	misspelled
(n|i -e)	143	aivotärähdyksiä:	
		aivo () tärähdyksen (n|i -e) ä	
		hoplofoobisia: hoplofoobisen (n|i -e) a	
(-n n|s)	138	aivokurkiaisen: aivokurkiainen (-n n|s) n	
(t|d)	97	häädöt: häätö (t|d) t	
(a|s -t)	83	amppeleissa: amppeleita (a|s -t) sa	
(ä|t -l)	82	näöltään: näöllä (ä|t -l) ään	
(-e -s)	77	esoteerinen: esoteerisen (-e -s) en	"inverse" of
(t|n)	75	abstrahoinnin: abstrahointi (t|n) n	
(a|t -l)	75	matkapuhelimeltaan:	
		matka () puhelimella (a|t -l) aan	

6. Semi-Supervised Learning of Morphological Analysis

In computational morphology, two starkly different approaches are well known, namely rule-based ones and methods based on unsupervised machine learning. The former approach produces output of excellent accuracy, but at a high cost of language-specific manual labor. In contrast, the unsupervised approach produces analyses whose quality are sufficient only in some applications, but once the method has been developed, adapting it to a new language requires a very limited manual effort. An open question for machine learning based approaches is: Given the existing methods, is it more cost-effective to develop better unsupervised methods or to annotate some amount of data and then employ it for training? We will attempt to answer this question by developing semi-supervised methods for morphological segmentation.

Formulating the problem in this fashion places certain constraints on the learning setting. In particular, we assume that manually annotated training sets will be small, since producing large, high-quality annotation is labor intensive. The learning method must, therefore, be able to derive as much benefit as possible from the little annotated data that is available. A similar focus on improving performance with a small amount of annotation effort have been explored for other natural language processing problems. For example, Garrette and Baldridge [2013] develop as good a part-of-speech tagger as possible given only two hours of annotation effort. In practice we will employ annotated sets on the order of 100 to $1,000$ word forms. If the annotation is given in the form of segmentation, annotating 100 words manually is a small effort, certainly requiring less than an hour for a native speaker.

Next, we specify our learning setting in more detail. The studied setting is a particular form of weakly supervised learning for morphological analysis. The learning system has at its disposal:

1. A small annotated training set typically containing no more than $1,000$ word types. The annotation is given in the form of a segmentation, and, in some cases, a full morphological analysis

2. A small annotated development set for hyperparameter optimization whose size is measured in the hundreds.

3. A large unannotated training set of words, containing hundreds of thousands or millions of word types

It is assumed that the supervision is provided only in the form of annotated words and no other sources of supervision are employed. This setting, therefore, differs from work employing alternative forms of supervision [Yarowsky and Wicentowski, 2000, Wicentowski and Yarowsky, 2003, Snyder and Barzilay, 2008b].

In this chapter, we will focus mainly on morphological segmentation, as developing and evaluating weakly supervised morphological segmentation methods is straightforward compared to what would be required for morphological analysis. In particular, even an unsupervised method can produce segmentations as output, and segmentations can be directly compared for accuracy. In contrast, the set of lemma and affix tags employed in morphological analysis cause problems both in training and evaluation. In particular, a machine-learning-based morphological analyzer can only assign the lemma and affix tags that occur in its training set. Assuming that the full set of lemmas and affixes must be seen in a small annotated set we study here is unrealistic. Alternatively, the morphological analysis task can be reformulated in some fashion that does not require the knowledge of the true lemma and affix set. An example of such a reformulation is the two-step approach presented in the context of modeling allomorphy in Section 5.2.1 which sidesteps the true lemma and affix set by focusing on the morphological relations between word-forms. Even this reformulated task, however, performs morphological segmentation in its first step, and is consequently more complicated. Moreover, evaluating the accuracy of a system that produces output in the correct tagset is straightforward, whereas evaluating any alternative output requires more complicated evaluation procedures. Although, several such evaluation methods exist, they are known to have different strengths and weaknesses [Virpioja et al., 2011]. This makes the evaluation a problem

in itself.

This chapter is structured as follows: We begin by reviewing the literature on semi-supervised morphological analysis in Section 6.1. In Section 6.2 we then discuss the work in Publication III where varying preprocessing decisions for word counts are evaluated in the context of unsupervised morphological segmentation. This work turns out to lead to an efficient hyperparameter formulation for controlling how much a Morfessor Baseline model segments. Then in Section 6.3 we present Semi-Supervised Morfessor, the morphological segmentation method originally introduced in Publication IV which builds on the hyperparameter optimization techniques in Publication III, extending them to the full semi-supervised setting. We then, in Section 6.4, depart into the study of semi-supervised morpheme labeling based on a morphological segmentation, following the work started in Publication IV, and further compare the performance of Semi-Supervised Morfessor to the state of art in the Morpho Challenge 2010 Competition 1 [Kurimo et al., 2010]. After this detour, we return to morphological segmentation in Section 6.5 where we change machine learning tools and switch mainly unsupervised techniques based on generative probabilistic models for supervised techniques and discriminative training. In particular we employ Conditional Random Fields [Lafferty et al., 2001], following Publication V and Publication VI. Finally, in Section 6.6 we provide a detailed empirical comparison of semi-supervised morphological methods following Publication VII.

6.1 Literature Review

In general, semi-supervised learning can be approached from two directions: On the one hand, one may improve an unsupervised method by adding some annotated data, or, on the other hand, improve a supervised method by leveraging the unannotated data. In morphological analysis, both approaches are possible. The techniques involved are, however, rather different [Zhu, 2006, Zhu and Goldberg, 2009, Daumé III, 2009]. In simplified terms, when there is very little annotated data, techniques based on unsupervised learning tend to perform best, as supervised techniques will only lead to overfitting. However, as the amount of annotated data increases, techniques based on supervised learning tend to surpass the unsupervised ones. The key question is at what number of annotated words this will occur. Intuitively, a morphological analyzer needs to be

able to segment many words containing morphs that cannot possibly be present in a small training set, and therefore unsupervised methods could be hypothesized to have an advantage in this task.

6.1.1 Weakly Supervised Training Setups

As discussed in the previous sections, we discuss the problem of morphological analysis in a setting where we have small annotated training and development sets, in addition to a large unannotated training set. Since the manual annotation effort required is small, we refer to this setting as *weakly supervised*. This setting excludes work where the supervision is extracted from some auxiliary source, such as a parallel corpus [Snyder and Barzilay, 2008a,b, Chahuneau et al., 2013]. Since these methods require a different experimental setting to evaluate, they will not be included in the further classification in this chapter.

In the studied weakly supervised setting, the supervision in the annotated training sets can be employed in several fashions. First, unsupervised methods may employ the annotated data to adjust some of their hyperparameters. While the term unsupervised learning itself suggests that such adjusting is infeasible, this type of tuning is nevertheless common [Creutz and Lagus, 2007, Çöltekin, 2010, Monson et al., 2010, Spiegler and Flach, 2010, Sirts and Goldwater, 2013]. In this chapter we will refer to unsupervised methods that do not adjust hyperparameters from annotated data as *unsupervised (USV)* methods. Meanwhile, we will refer to methods that do employ the annotated data to optimize their hyperparameters as *unsupervised methods with hyperparameter tuning (PSV)*. It should be noted that the boundary between these method categories is not always clear, as several unsupervised methods employ internal hyperparameters whose values were originally set by the method authors based on the output of the algorithm. In that case, no explicit annotated data was used for hyperparameter tuning, but implicitly the linguistic knowledge of the authors was, nevertheless, employed. Second, the annotated data sets can be applied for training while ignoring the unannotated words. We will refer to this as *supervised (SV)* learning. Third, methods that utilize both the annotated data sets as well as the unannotated data sets are referred to as *semi-supervised (SSV)* methods.

6.1.2 Classification of Weakly Supervised Morphological Segmentation Methods

In this section, we characterize semi-supervised methods proposed in the literature. As unsupervised methods are well-known and have been reviewed by several authors [Hammarström and Borin, 2011, Creutz and Lagus, 2007, Goldsmith, 2001], we will not discuss these in much detail. It should be mentioned, however, that most unsupervised methods can easily employ annotated data for hyperparameter optimization. In Section 6.2 we further show that a method with no explicit hyperparameters can instead utilize data selection schemes to similar effect. Therefore, in principle, any unsupervised method can benefit from annotated data.

An alternative approach is to employ only the annotated data in a supervised setting. Such work includes the method of Eger [2013] who performs supervised segmentation by exhaustive enumeration with a generative Markov model on morphs. Similarly, our work in Publication V evaluates the segmentation performance of a Conditional Random Field trained solely on the annotated data. Supervised approaches are, however, limited to generalizing the phenomena in the annotated data. We assume that the annotated training data is small and this implies that many of the morphs we would like to analyze will be unknown to any purely supervised model.

We now discuss specifically methods that employ the annotated data in a semi-supervised fashion. Such methods have been devised starting from either an unsupervised or a supervised method. Methods that begin from an unsupervised model and are then extended to semi-supervised learning include the log-linear model of Poon et al. [2009], semi-supervised Morfessor introduced in Publication IV, the generative model Promodes for letter transitions and boundaries [Spiegler and Flach, 2010], the Hidden Markov Model approach of Kılıç and Bozsahin [2012], the extension of Adaptor Grammars (AG) [Johnson et al., 2007] to semi-supervised learning [Sirts and Goldwater, 2013], and, finally, Morfessor FlatCat by Grönroos et al. [2014]. In contrast, methods approaching from the supervised direction includes our work in Publication V and Publication VI (Section 6.5). We will now characterize these methods according to their similarities and differences.

Learning Lexicons versus Detecting Boundaries We begin by dividing the methods described above into two categories: **lexicon-based** [Poon et al.,

2009, Kılıç and Bozsahin, 2012, Eger, 2013, Sirts and Goldwater, 2013, Grönroos et al., 2014], including Publication IV, and **boundary detection** [Harris, 1955, Spiegler and Flach, 2010] as well as Publication V. In the former, the model learns lexical units, whereas in the latter the model learns properties of morph boundaries. For example, in the case of Morfessor [Creutz et al., 2007] the lexical units correspond to morphs while in AGs [Sirts and Goldwater, 2013] the units are parse-trees. Meanwhile, consider the CRF approach of Publication V, Publication VI, as well as the classical approach of Harris [1955] which do not store explicit morph-like units, but instead identify morph boundary positions utilizing substring contexts and letter successor varieties, respectively. In general, whether it is easier to discover morphs or morph boundaries is largely an empirical question. So far there have been no models that would combine both approaches, but this could be an interesting future development.

Generative versus Discriminative Learning The second main distinction divides the models into **generative** and **discriminative** approaches. The generative approaches [Poon et al., 2009, Spiegler and Flach, 2010, Kılıç and Bozsahin, 2012, Eger, 2013, Sirts and Goldwater, 2013, Grönroos et al., 2014] and Publication IV model the joint distribution of words and their corresponding segmentations, whereas discriminative approaches, including that of Harris [1955], Publication V, and Publication VI, directly estimate a conditional distribution of segmentation *given* a word. In other words, whereas generative methods generate both words and segmentations, the discriminative methods generate only segmentations given words. The generative models are naturally applicable for unsupervised learning. Meanwhile, discriminative modeling always requires some annotated data. Lastly, it appears that most lexicon-based methods are generative and most boundary detection methods are discriminative. However, it should be pointed out that this is a trend rather than a rule, as exemplified by generative boundary detection method of Spiegler and Flach [2010].

Semi-Supervised Learning Approaches Both generative and discriminative models can be extended to utilize annotated as well as unannotated data in a semi-supervised manner. The applicable techniques, however, differ. For generative models, semi-supervised learning is in principle trivial: for the labeled words, the segmentation is fixed to its correct value, as exemplified by the approaches of Poon et al. [2009], Spiegler and Flach [2010], Sirts and Goldwater [2013]. Meanwhile, the semi-supervised set-

ting also makes it possible to apply discriminative techniques to generative models. In particular, model hyper-parameters can be selected to optimize segmentation performance rather than some generative objective, such as likelihood. Special cases of hyper-parameter selection include the weighted objective function in Publication IV and [Grönroos et al., 2014], data selection in Publication III and by Sirts and Goldwater [2013], as well as grammar template selection [Sirts and Goldwater, 2013]. As for the weighted objective function and grammar template selection, the weights and templates are optimized to maximize segmentation accuracy on a held out development set. Meanwhile, data selection is based on the observation that omitting some of the training data can improve segmentation accuracy.

For discriminative models, a straightforward semi-supervised learning technique is adding features derived from the unlabeled data, as exemplified by the CRF approach in Publication VI. However, discriminative semi-supervised learning is in general a much researched field with numerous, diverse techniques [Zhu and Goldberg, 2009]. For example, for the CRF model alone, there exist several proposed semi-supervised learning approaches [Jiao et al., 2006, Mann and McCallum, 2008, Wang et al., 2009].

On Local Search In what follows, we will discuss a potential pitfall of some algorithms which utilize local search procedures in the parameter estimation process, as exemplified by the Morfessor model family [Creutz et al., 2007]. As discussed in Section 4.1.1, the Morfessor algorithm finds a local optimum of its objective function using a local search procedure. This complicates model development because if two model variants perform differently empirically, it is uncertain whether the difference is caused by the model or the estimation method, as discussed also by Goldwater [2006, Section 4.2.2.3]. Therefore, in contrast, within the adaptor grammar framework [Johnson et al., 2007, Sirts and Goldwater, 2013], the focus has not been on finding a single best model, but rather to marginalize over model parameters to find the posterior distribution over segmentations of the words. Another approach to the problem of bad local optima is to start a local search near some known good solution. This approach is taken in Morfessor FlatCat, for which it was found that initializing the model with the segmentations produced by the supervised CRF model (with a convex objective function) yields improved results [Grönroos et al., 2014].

6.2 Utilizing Training Set Word Frequencies as Implicit Hyperparameters

In this section we will begin the exploration of generative models for semi-supervised morphological segmentation, following Publication III. The purpose is two-fold. First, to examine properties of unsupervised models. Second, to discover efficient implicit hyperparameters for semi-supervised learning. As reviewed in Section 6.1.1, hyperparameter adjustment adapts the inductive bias of an unsupervised method which can yield improvements in performance. All models do not have appropriate hyperparameters, however.

Despite the overarching theme of semi-supervised learning, the main focus of this section is to understand unsupervised learning and how to preprocess the training data appropriately. An important property of natural language is the power-law distribution of words [Zipf, 1932]. This very skewed distribution affects the training of probabilistic models and, consequently, when training an unsupervised model, one needs to decide how to preprocess these word frequencies. First, one must decide whether training is based purely on types, that is disregarding the word counts, or on tokens, that is employing the word counts directly. We will also examine a variant in between these extremes, namely employing the logarithm of the token count. Second, we will examine the effect of filtering out words that occur rarely. When discussing the results of Allomorfessor in Chapter 5.4.1, a small side observation was that the morphological segmentation performance of Morfessor Baseline improved with the elimination of words occurring only once in the training corpus. In this section, we will experiment with this issue in more detail.

Previous work report that training on tokens and types can affect the segmentation results [Creutz and Lagus, 2004, 2005b, Goldwater et al., 2006, Poon et al., 2009]. Creutz and Lagus [2004, 2005b, 2007] report that Morfessor Baseline tends to under-segment, that is, segment much less than the gold standard, when trained on tokens. However, training on word types alleviates the problem. This is less of a problem for Morfessor Categories-MAP, however, for English, Creutz and Lagus [2007] report that recall is lowered with increasing training data. Such results may be caused by increasing amount of noise, such as misspellings and foreign words, in the larger word lists. However, there may also be other, model-internal reasons. Poon et al. [2009] report better results when

training their log-linear model on types rather than tokens. Goldwater et al. [2006] demonstrate that, for a certain class of models, the generation of the token frequencies can be separated into a separate second-level model component, such that, depending on a hyper parameter, the first-level model is trained on what can be seen as a continuum between tokens and types. They find experimentally that their morphological segmentation model performs best when the parameter value is closer to utilizing types, whereas utilizing tokens leads to under-segmentation.

In summary, there are several reports of different models performing better on types than tokens. Moreover, there are also reports where larger training sets cause reduced performance. Naturally, one would expect that with more data the performance should converge towards some asymptotic upper bound, and, therefore, this behavior is surprising. The models that behave in this fashion have in common that they apply automatic model selection techniques to determine how much the method should segment, based on the training data. An open question is how this mechanism interacts with the choice of types or tokens. A further question is why the filtering of rare words helped Morfessor Baseline in the previous chapter. It could be hypothesized that employing tokens for training as well as filtering the least frequent forms in the corpus would be an efficient manner of filtering out the effect of noise words, such as misspellings and foreign words, as these would tend to be infrequent, and could, therefore, be overemphasized by training based on types. When considering semi-supervised learning, a downside to Morfessor Baseline is that it does not have any hyperparameters that could be adjusted based on annotated training data. In this section, we will develop a hyperparameter that explicitly controls the amount of segmentation that the method performs. Here, we will do this in the interest of controlling the experimentation, but in Section 6.3 we will employ it in the full semi-supervised setting.

6.2.1 Analysis of the Effects of Frequency

Consider a generative latent variable model that generates words independently. Assume we are looking for maximum posterior (MAP) parameters. Such models include Morfessor Baseline [Creutz and Lagus, 2002, 2007] and Morfessor Categories-MAP [Creutz and Lagus, 2005a]. We can write the optimization problem as follows:

$$\hat{\boldsymbol{\theta}}_{MAP} = \arg\max_{\boldsymbol{\theta}} p(\mathcal{U}|\boldsymbol{\theta})p(\boldsymbol{\theta}) = \prod_{j=1}^{W} p(\mathbf{x}^{(i)}|\boldsymbol{\theta})p(\boldsymbol{\theta}) \tag{6.1}$$

If we train on tokens, then $\mathcal{U}$ contains the same word type $\mathbf{s}_j$ repeatedly. Let C_j denote the number of times the word $\mathbf{s}_j$ occurs in $\mathcal{U}$. Because of the independent generation of words the marginal likelihood for all instances of the word $\mathbf{s}_j$ is the same. Consequently, the likelihood can be factored by word type $\mathbf{s}_j$ as:

$$\hat{\boldsymbol{\theta}}_{MAP} = \arg\max_{\boldsymbol{\theta}} \prod_{j=1}^{V} p(\mathbf{x}^{(i)} = \mathbf{s}_j|\boldsymbol{\theta})^{C_j} p(\boldsymbol{\theta}), \tag{6.2}$$

where V is the number of word types in $\mathcal{U}$. Based on this factorization, we can create different experimental conditions by defining a transform $f(C_j)$ on the word counts. For example, $f(C_j) = 1$ and $f(C_j) = C_j$ correspond to training with types and tokens, respectively. Generally, we will experiment with transforms of the following form:

$$f(C_j) = \begin{cases} 0 & \text{if } C_j < T \\ \alpha g(C_j) & \text{otherwise} \end{cases} \tag{6.3}$$

where T is a frequency cutoff threshold and $g(C_j)$ is a function that transforms the counts. If $\alpha = 1$ and $g(C_j) = 1$ we train on word types; if $\alpha = 1$ and $g(C_j) = C_j$, we train on tokens. In addition, we will employ the logarithmic function $g(C_j) = \ln(1 + C_j)$. The frequency threshold T can be used for pruning rare words from the training data.

With this parameterization, we can rewrite the optimization problem into the form:

$$\hat{\boldsymbol{\theta}}_{wMAP} = \arg\max_{\boldsymbol{\theta}} \prod_{j=1}^{V} p(\mathbf{x}^{(i)} = \mathbf{s}_j|\boldsymbol{\theta})^{f(C_j)} p(\boldsymbol{\theta}), \tag{6.4}$$

where $wMAP$ indicates that this is not the true MAP-estimate, but rather one which has been modified by weighting.

Since the logarithm is a monotonic function, we can equivalently optimize the logarithm:

$$\hat{\boldsymbol{\theta}}_{wMAP} = \arg\max_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}) + \sum_{j=1}^{V} f(C_j) \log p(\mathbf{x}^{(i)} = \mathbf{s}_j|\boldsymbol{\theta}) \tag{6.5}$$

$$= \arg\max_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}) + \sum_{j \in \{C_j \geq T\}} \alpha g(C_j) \log p(\mathbf{x}^{(i)} = \mathbf{s}_j|\boldsymbol{\theta}) \tag{6.6}$$

$$= \arg\max_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}) + \alpha \sum_{j \in \{C_j \geq T\}} g(C_j) \log p(\mathbf{x}^{(i)} = \mathbf{s}_j|\boldsymbol{\theta}) \tag{6.7}$$

The other parts of this expression are previously suggested manipulations of the word counts, but not the parameter α, which we introduce in order to control the experimental setting. It can be seen that the parameter α operates as a global weight between the likelihood and the prior.

As discussed in Section 4.1.1, for Morfessor Baseline the likelihood term prefers analyses with few morphs, whereas the prior term prefers lexicons with few morphemes. These opposing terms then balance each other and produce a compromise. The parameter α then affects this balance, and can intuitively be thought of as a lever that controls how much the model segments. In the next section we will demonstrate this behavior experimentally. One can further notice that the transformed counts $g(C_j)$ affect the likelihood but not the prior. Therefore, when employing token-based training, the balance between the two will be different than when training with types. With the introduction of the parameter α we can adjust the amount of segmentation separately, and, therefore, learn whether the previously reported superiority of type-based training, or merely, by its effect on the balance between prior and likelihood. Finally, it can be seen that the threshold T also affects the balance between prior and likelihood by filtering out words on the likelihood side. This leads to more segmentation as a result of filtering out words from the data set.

6.2.2 Experiments

We perform experiments on the English and Finnish parts of the Morpho Challenge 2009 data set [Kurimo et al., 2009c], and we apply the Morpho Challenge Competition 1 evaluation measure, presented in Section 2.2.2. For tuning the parameters of the weight function, we sampled a development set that did not contain any of the words in the final test set. The development set included $2,000$ words for English and $8,000$ words for Finnish. The languages were selected based on their different morphological characteristics. The experiment setup is as follows: We consider the following set of functions of the counts $g(C_i)$: constant (types), linear (tokens), and logarithmic. To control for the effect of the balance between the prior and likelihood we optimized the cutoff threshold T and the weight parameter α by choosing values that gave the optimal F-measure on the development set.

We apply our own re-implementation of Morfessor Baseline which is based on the Morfessor 1.0 software [Creutz and Lagus, 2005b]. The format of the input data is a list of words and their counts, so the function $f(C_i)$ is, in principle, trivial to apply as preprocessing. However, because the Morfessor prior assumes integer counts, the parameter α was implemented as a global weight for the likelihood. Otherwise, the training data was modified according to the respective function before training. The re-

sult of the logarithmic function was rounded to the nearest integer. In the prior $p(\boldsymbol{\theta})$, we apply implicit morph length and frequency priors. The model parameters are then estimated with the standard parameter estimation algorithm, presented in Section 4.1.1. Subsequent to training, the segmentations for the development and test set were found by calculating the best segmentation according to the model parameters, allowing new morphs with the approximate cost of adding them into the morph lexicon, as described in more detail in Section 4.1.1.

6.2.3 Results

We trained Morfessor Baseline with the frequencies modified according to the constant, linear, and logarithmic function, where the two former ones correspond to training on word types and tokens, respectively, while the third is an intermediate form of the first two. We applied grid search to optimize the cutoff threshold parameter T and the weight parameter α, selecting as optimal the values that yield the highest F-measure on the development set. When $\alpha = 1.0$ and $T = 1$, the training corresponds to standard Morfessor Baseline.

Figures 6.1 and 6.2 show the precision-recall curves when varying only one of the parameters separately for English and Finnish, respectively. Varying only T removes infrequent words. Meanwhile, varying α keeps the training set word list unchanged while adjusting the weight on the prior. It can be seen that reducing α increases recall regardless of how the word frequencies have been preprocessed. Interestingly, increasing T improves recall to a similar degree for the constant function, but not for the logarithmic and linear functions.

Table 6.1 compares the results when optimizing both T and α to a baseline where neither is optimized. It can be seen universally that the baseline suffers from under-segmentation which is further worsened when applying the logarithmic or linear counts. For the case where the parameters are optimized we can note that there are large improvements over baseline. Furthermore, precision and recall are more evenly balanced, indicating that the optimization of the parameters balances precision and recall. Next, we consider the relative performance of the count modifying functions. For English, logarithmic counts produced the highest F-measure, but the difference to that of the constant function was not statistically significant according to the Wilcoxon signed-rank test. Linear counts produced clearly inferior results. For Finnish, the constant func-

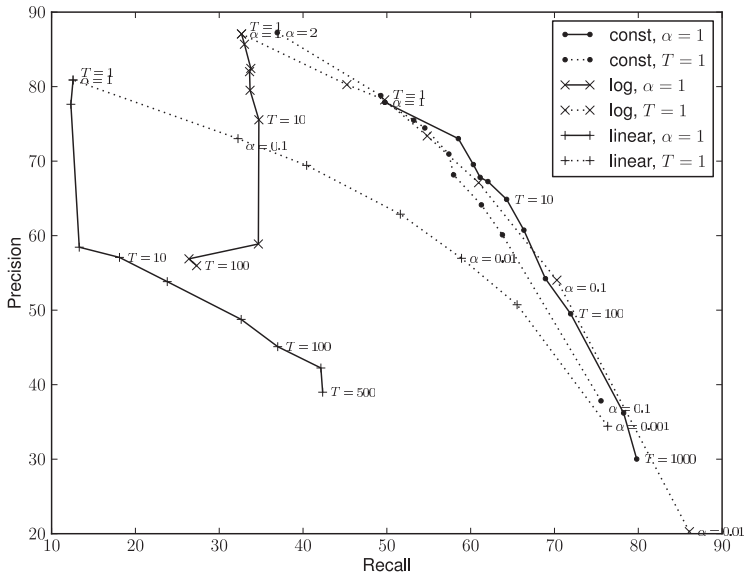

Figure 6.1. Precision-recall curves for English with constant (const), logarithmic (log), and linear frequency function types and varying function parameters α or T. Solid lines, $\alpha = 1$ and T varied; dashed lines, $T = 1$ and α varied. F-scores improve towards top-right.

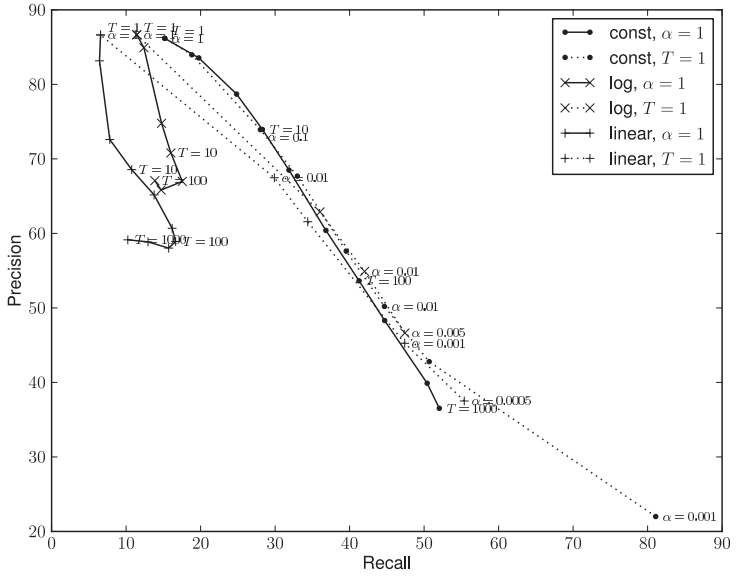

Figure 6.2. Precision-recall curves for Finnish with constant (const), logarithmic (log), and linear frequency function types and varying function parameters α or T. Solid lines, $\alpha = 1$ and T varied; dashed lines, $T = 1$ and α varied. F-scores improve towards top-right.

tion was slightly but significantly better than the logarithmic and linear functions.

The optimal parameter values can be analyzed as follows: Regarding the weight parameter α, the constant function prefers values close to the default value 1 for English. However, for Finnish this is not the case, and all functions benefit from $\alpha < 1$. Generally, the linear function requires the smallest α. Meanwhile, the cutoff thresholds $T > 1$ are beneficial for English, except in combination with the linear function. In contrast, for Finnish the optimal threshold is invariably $T = 1$.

Function	*Optimized*	*T*	α	*Pre*	*Rec*	*F-m*
English						
constant	no	1	1	76.13	48.97	**59.60**
logarithmic	no	1	1	87.76	31.77	46.65
linear	no	1	1	84.93	12.00	21.03
constant	yes	10	1.1	62.04	62.27	62.16
logarithmic	yes	20	0.2	57.85	67.62	**62.35**
linear	yes	1	0.01	53.96	56.42	55.16
Finnish						
constant	no	1	1	89.50	15.70	**26.72**
logarithmic	no	1	1	91.24	11.95	21.13
linear	no	1	1	91.82	6.75	12.57
constant	yes	1	0.01	53.77	45.16	**49.09**
logarithmic	yes	1	0.01	57.87	42.06	48.72
linear	yes	1	0.001	48.86	47.37	48.10

Table 6.1. Precision (pre), recall (rec) and F-measure (F-m) on the final test set with the different function types for word frequencies. In optimized cases (opt), T and α are selected according to the best F-measure for the development set.

6.2.4 Discussion

Previous results indicate that training Morfessor Baseline on word types yields better performance than utilizing word tokens [Creutz and Lagus, 2004, 2005b, 2007], with similar results reported for other models [Goldwater et al., 2006, Poon et al., 2009]. It is, however, not clear from previous work what causes the deterioration in performance when training on tokens. Intuitively, when utilizing word frequency information, the learning method has more information at its disposal and should, conse-

quently, be able to perform better. The previous results can be interpreted either as the word frequency being inherently of little benefit for this task, or alternatively, the deterioration in performance is caused by some particular properties of the employed models. Given the presented analysis of how the word frequencies affect Morfessor Baseline, we can hypothesize, that previous results were merely a result of under-segmentation, because training on tokens tends to produce more sparse segmentation than the linguistic analysis would prefer. If this is the case, and frequency information is inherently useful, we would expect to see improvement in performance for the linear and logarithmic function over the constant one when the weight α and the threshold T are optimal. The results indicate that this is not the case for Morfessor Baseline. In the results by Goldwater et al. [2006] it was found that the optimal interpolation between types and tokens employed mostly types, but also tokens to a smaller extent. Therefore, it may be the case that performance benefits could be found with some other intermediate form of types and tokens than the logarithmic function. However, discovering such an intermediate form requires further work. Both our work and the work by Goldwater et al. [2006], Poon et al. [2009] suggest that training on word types is a good pragmatic default option.

From a pragmatic perspective, optimizing the weight parameter α and cutoff threshold parameter T yielded large performance improvements. They are, therefore, promising hyperparameters to employ in the case where one has some annotated data for hyperparameter tuning. Several other hyperparameters have been suggested in the literature [Creutz and Lagus, 2007, Çöltekin, 2010, Spiegler and Flach, 2010, Sirts and Goldwater, 2013]. Such techniques are discussed in more detail in Sections 6.1 and 6.3. The cutoff threshold parameter T did improve performance for English, indicating that there is indeed a benefit to removing infrequent forms. Therefore, it may be the case that the infrequent words in the English training set are mostly noise, for example, misspellings or foreign words that do not convey useful information about the morphology of the language. However, for Finnish, removing infrequent forms was not beneficial. A possible reason is that in an agglutinative language, such as Finnish, many valid inflected forms are very rare and, therefore, these valid words may make up a considerable part of the removed infrequent words. It can be concluded that whether optimizing the cutoff threshold T is beneficial is language specific. In contrast, optimizing the weight

pattern α improved performance for all frequency functions and both languages.

6.3 Semi-Supervised Morfessor

In the previous section it was found that Morfessor Baseline results can be improved greatly by adjusting hyperparameters that affect how much the model segments. In this Section, we formulate a further extension of Morfessor Baseline to semi-supervised learning. Particularly, in addition to the unannotated data, we have a small annotated data set that we utilize for training.

6.3.1 Semi-Supervised Training of Morfessor Baseline

Morfessor Baseline is a generative probabilistic model with a latent segmentation variable. In principle, semi-supervised training for such a model is straightforward: The segmentations of the annotated training examples are fixed to their correct values, and standard latent variable training, usually Expectation-Maximization (EM) is employed on the unannotated data.

In the semi-supervised setting we train on both the unannotated data $\mathcal{U}$ and the annotated data $\mathcal{D}$. We index the data set, such that the unannotated data is given by $\mathcal{U} = \{\mathbf{x}^{(i)}\}_{i=1}^{W}$, and the annotated data occupies different indices, $\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=W+1}^{W+L}$. We can then write the optimization problem as follows:

$$\hat{\boldsymbol{\theta}}_{MAP} = \arg\max_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}) + \sum_{i=1}^{W} \log \sum_{\mathbf{z}^{(i)} \in \mathrm{SEG}(\mathbf{x}^{(i)})} p(\mathbf{x}^{(i)}|\boldsymbol{\theta}, \mathbf{z}^{(i)}) p(\mathbf{z}^{(i)}) + \sum_{i=W+1}^{W+L} \log p(\mathbf{x}^{(i)}|\mathbf{z}^{(i)} = \boldsymbol{y}^{(i)}, \boldsymbol{\theta}) \quad (6.8)$$

We already noted in Section 4.1.1 that applying EM cannot be applied to Morfessor Baseline. Instead, one could fix the annotated examples, and then employ the standard Morfessor Baseline parameter estimation method. This corresponds to concentrating all the mass of $p(\mathbf{z})$ to a single segmentation $\hat{\mathbf{Z}}$, and optimizing to find the best segmentation of the input

data set. The optimization problem is given by:

$$\hat{\boldsymbol{\theta}}_{BL_F} = \arg\max_{\boldsymbol{\theta},\hat{\mathbf{Z}}} \log p(\boldsymbol{\theta}) + \sum_{i=1}^{W} \log p(\mathbf{x}^{(i)}|\boldsymbol{\theta}, \hat{\mathbf{z}}^{(i)}) + \sum_{i=W+1}^{W+L} \log p(\mathbf{x}^{(i)}|\mathbf{z}^{(i)} = \boldsymbol{y}^{(i)}, \boldsymbol{\theta}) \quad (6.9)$$

Preliminary results, however, indicated that this approach leads to very small or nonexistent improvements. Analyzing Expression 6.9 reveals that if $L << W$, then the annotated data has very little effect on the objective function. As we assume a setting with a large unannotated word list and a small annotated set, the above holds. To correct for this we introduce a new weight parameter β to increase the effect of the annotated set. This parameter is to be adjusted based on development set performance. Moreover, in Section 6.2, we found that adjusting hyperparameters related to how much the model segments on average, namely the weight parameter α and the frequency cutoff threshold T, yielded large performance improvements. Because of the language specificity of the frequency cutoff threshold T, we choose to only include the weight parameter α to form a new objective function. This results in what we refer to as Semi-Supervised Morfessor with the following optimization problem:

$$\hat{\boldsymbol{\theta}}_{BL_{\alpha\beta}} = \arg\max_{\boldsymbol{\theta},\hat{\mathbf{Z}}} \log p(\boldsymbol{\theta}) + \alpha \sum_{i=1}^{W} \log p(\mathbf{x}^{(i)}|\boldsymbol{\theta}, \hat{\mathbf{z}}^{(i)}) + \beta \sum_{i=W+1}^{W+L} \log p(\mathbf{x}^{(i)}|\mathbf{z}^{(i)} = \boldsymbol{y}^{(i)}, \boldsymbol{\theta}) \quad (6.10)$$

It should be noted that this weighted objective function no longer corresponds to a generative model. The training procedure is therefore a mixture of generative training and discriminative training. In practice, a grid search is performed where α and β are optimized for development set performance. For each value of (α, β) the standard Morfessor Baseline parameter estimation method is employed, together with fixing the segmentations of the annotated data to their known values. Since the training data we employ contains several alternative segmentations for some word forms, in practice the algorithm also updates its current segmentation $\mathbf{z}^{(i)}$ for the annotated examples, however, only among the alternative segmentations $\boldsymbol{y}^{(i)} \in \mathcal{Y}^{(i)}$, where $\mathcal{Y}^{(i)}$ denotes the set of alternative segmentations given in the annotated training data for word $\mathbf{x}^{(i)}$.

6.3.2 Experiments

We employ the same data set as in Section 6.2.2, and apply the novel methods to the English and Finnish parts of the Morpho Challenge 2009 data set. We evaluate with the Morpho Challenge Competition 1 evaluation measure, presented in Section 2.2.2. We independently sample from the Morpho Challenge evaluation data a training, development and test set. The training set contains 100 to 10 000 words, the development set has 500 words and the test set contains 10,000 words for English and 200,000 words for Finnish.[1] Following the results in Section 6.2 we train on word types, ignoring word counts in the training data, and setting the word count to 1 for each training set word.

We compare the semi-supervised Morfessor method with the following baselines: Unsupervised Morfessor Baseline and Unsupervised Morfessor Baseline with the weight parameter α optimized. We report semi-supervised results for the data set sizes 100, 1,000, and 10,000. We also report the results when weighting is not employed, but the latent segmentations are fixed the to the training data as in Expression (6.9). This is equivalent to setting $\alpha = \beta = 1$ in Expression (6.10). Finally, we report the results when optimizing both α and β.

6.3.3 Results

The experimental results with Morpho Challenge Competition 1 evaluation metric's precision, recall and F1 are shown in Table 6.2 together with the optimal hyperparameter values for Semi-Supervised Morfessor compared with the standard unsupervised Morfessor Baseline. Despite a much smaller development set, we can see that, similarly to the results in Section 6.2, the optimization of the weight parameter α brings a small performance improvement for English, and a very large one for Finnish. Employing semi-supervised learning by merely fixing the annotated data segmentations without adjusting the hyperparameters α and β improves the F1-score to a small degree. In contrast, when optimizing both hyper-

[1] Publication IV describes the procedure such that there would be no overlap between the training and test set. Because of a programming error discovered at the time of writing this, the sets were independently sampled with overlap. The biggest training set and the test set contain 16% overlap for English, and 1.6% for Finnish. This does, however, not invalidate the relative results, since the overlap is partial. Moreover, we will evaluate this algorithm thoroughly in Sections 6.4 and 6.6.

Method	Opt	α	β	Train (ann.)	Pre.	Rec.	F1
English							
MORFESSOR BL (USV)	no	1	-	0	74.63	50.08	59.94
MORFESSOR BL (PSV)	α	0.75	-	0	68.48	55.07	61.04
MORFESSOR BL (SSV)	no	1	1	100	74.59	50.01	59.88
MORFESSOR BL (SSV)	no	1	1	1,000	75.13	50.37	60.31
MORFESSOR BL (SSV)	no	1	1	10,000	79.12	50.61	61.73
MORFESSOR BL (SSV)	α, β	0.75	750	100	67.82	62.74	65.18
MORFESSOR BL (SSV)	α, β	1	500	1,000	69.72	66.92	68.29
MORFESSOR BL (SSV)	α, β	1.75	175	10,000	77.35	68.85	**72.86**
Finnish							
MORFESSOR BL (USV)	no	1	-	0	89.52	15.7	26.72
MORFESSOR BL (PSV)	α	0.01	-	0	53.72	45.16	49.07
MORFESSOR BL (SSV)	no	1	1	100	89.56	15.69	26.70
MORFESSOR BL (SSV)	no	1	1	1,000	89.56	15.7	26.72
MORFESSOR BL (SSV)	no	1	1	10,000	89.74	15.68	26.70
MORFESSOR BL (SSV)	α, β	0.01	500	100	50.67	54.81	52.66
MORFESSOR BL (SSV)	α, β	0.05	2,500	1,000	61.03	52.38	56.38
MORFESSOR BL (SSV)	α, β	0.1	500	10,000	69.14	53.4	**60.26**

Table 6.2. Results of the Semi-Supervised Morfessor for Morpho Challenge 2009 data on English and Finnish with various amounts of training data and training protocols. The field *Opt* denotes whether, and which hyperparameters were adjusted – *no* corresponds to the standard unsupervised Morfessor Baseline, α to optimizing only the hyperparameter α, and α, β to the full Semi-Supervised Morfessor.

parameters the performance improves considerably for both languages.

6.3.4 Discussion

The presented experiments show considerable improvements for both languages. However, the large relative improvements for Finnish are mainly explained by the standard Morfessor Baseline method segmenting too sparsely for Finnish and the adjustment of the weight hyperparameter α correcting for this. Meanwhile, the standard approach to semi-supervised learning for generative models, merely fixing the segmentations of the annotated data, performed disappointingly. In contrast, when placing more weight on the annotated data via the parameter β, the performance improves considerably. It appears that the studied setting, where the number of annotated words is a very small fraction of the number of unannotated words, is challenging for generative training. In the absence of the weight parameter β on the annotated data, the unsupervised part of the data dominates the outcome of training.

The employed α weighting is in principle not specific to Morfessor Baseline, but can be employed for any model where the prior penalizes excessive segmentation. The β weighting is even more generally applicable, as it only requires that the likelihood can be factored into separate parts for the annotated and unannotated training data. Interestingly, it turned out that while Morfessor Categories-MAP employs a similar model structure to Morfessor Baseline, its hierarchical lexicon is problematic for the α weighting. The problems are detailed in [Grönroos et al., 2014] who propose removing the hierarchical lexicon from Categories-MAP and employing the α and β weighting to the resulting model, which is known as Morfessor FlatCat.

6.4 Semi-Supervised Morfessor in Morpho Challenge 2010

The experiments in the previous section show that supervision can improve results considerably. However, the results have not yet been compared to the state of the art. We noted that the experiments presented in the previous section, unfortunately, contained a mistake in the training data generation, such that training and test data partially overlapped. The results can, therefore, be suspected to be optimistic. To better assess the method's performance, we discuss the results of Semi-Supervised Mor-

fessor in the Morpho Challenge 2010 competition [Kurimo et al., 2010].

In these experiments the output of semi-supervised Morfessor was further augmented by a labeling phase. The annotated data was utilized to learn a hidden Markov model that annotates the segmentation into a labeling. The details of the labeling procedure are given in [Kohonen et al., 2010].

Morpho Challenge 2010 provides an annotated training set of $1,000$ words. We include only English, Finnish, and Turkish for for which gold standard segmentations are available. The development set sizes are 694, 835, and 763 for English, Finnish, and Turkish, respectively.

We gather the results of the top 5 methods in Competition 1, where performance is measured by the similarity to a linguistic gold standard analysis. For comparison, we also extract the result of Semi-Supervised Morfessor when only adapting the weight parameter α, that is only employing the development set for hyperparameter optimization while ignoring the training set. We collect the results from the tables containing all Morpho Challenges from 2007-2010.[2] The methods have been trained on the data from the year of their submission, but have all been re-evaluated utilizing the evaluation of Morpho Challenge 2010. We omit methods that submit alternative analyses, as it was demonstrated by Spiegler and Monson [2010] that the Morpho Challenge evaluation measure is unreliable when providing alternative analyses. Table 6.3 shows the collected results. The methods include Semi-Supervised Morfessor with α and β adjusted (MORFESSOR BL (SSV)), as well as in combination with the labeling scheme on top as presented in [Kohonen et al., 2010] (MORFESSOR BL+LAB (SSV))). The methods BERNHARD 1 (USV) and BERNHARD 2 (USV) are presented in [Bernhard, 2008]; LIGNOS BASE INF. (USV) in [Lignos, 2010]; and PARAMOR-MORF. UNION (USV), PARAMOR-MORF. MIMIC (USV), and PARAMOR MIMIC (USV) in [Monson et al., 2010].

From the results in Table 6.3 it can be seen that Semi-Supervised Morfessor performs better than any proposed unsupervised methods, as is to be expected. However, the difference to the best unsupervised methods is not very large.

[2] Available at: http://research.ics.aalto.fi/events/morphochallenge/

#	Method	Type	Train (ann.)	Pre.	Rec.	F1
English						
1.	MORFESSOR BL (SSV)	seg	1,000	65.62	69.28	67.40
2.	MORFESSOR BL+LAB (SSV)	lab	1,000	67.87	66.43	67.14
3.	BERNHARD 2 (USV)	lab	0	67.42	65.11	66.24
4.	BERNHARD 1 (USV)	lab	0	75.61	57.87	65.56
5.	LIGNOS BASE INF. (USV)	lab	0	80.77	53.76	64.55
...						
10.	MORFESSOR BL (PSV)	seg	0	60.33	59.55	59.94
Finnish						
1.	MORFESSOR BL+LAB (SSV)	lab	1,000	58.38	63.35	60.76
2.	MORFESSOR BL (SSV)	seg	1,000	57.59	55.21	56.38
3.	BERNHARD 2 (USV)	lab	0	63.92	44.48	52.45
4.	PARAMOR-MORF. UNION (USV)	seg	0	47.89	50.98	49.39
5.	MORFESSOR BL (PSV)	seg	0	56.97	42.98	49.00
Turkish						
1.	MORFESSOR BL+LAB (SSV)	lab	1,000	71.69	59.97	65.31
2.	MORFESSOR BL (SSV)	seg	1,000	65.71	47.15	54.90
3.	PARAMOR-MORF. MIMIC (USV)	seg	0	48.07	60.39	53.53
4.	PARAMOR-MORF. UNION (USV)	seg	0	47.25	60.01	52.88
5.	PARAMOR MIMIC (USV)	seg	0	49.54	54.77	52.02
...						
10.	MORFESSOR BL (PSV)	seg	0	40.71	46.76	43.52

Table 6.3. Morpho Challenge 2010 top methods according to Competition 1 after omitting competitors that submit alternative analyses. The column *Type* shows the output type of the method, where *seg* is segmentation and *lab* is labeling.

6.4.1 Discussion

Finally, we can return to the question at the beginning of this chapter: If we want to improve performance, is it more cost effective to devise better unsupervised methods or should we rather annotate some data? First, we must consider how large an annotated set can be considered cost effective. Since manually segmenting $1,000$ words can be done in a matter of hours, it appears to be both feasible and cost effective. The Morpho Challenge 2010 comparison in the previous setting can then give us some idea of whether it is better to annotate data, rather than improve an unsupervised method. It can be seen that Semi-Supervised Morfessor outperforms the best unsupervised methods. However, the margin is not very large. A big confounding factor in the comparison is whether the method outputs a segmentation or a morph labeling. Generally, however, we may conclude that semi-supervised learning enables a simple model, such as Morfessor Baseline, to perform very well, given on the order of $1,000$ training words.

6.5 Morphological Segmentation with Conditional Random Fields

In the previous sections, we have discussed generative probabilistic models created for the purpose of morphological segmentation. In general, many other segmentation tasks also exist within natural language processing. As a result, general methods for segmentation problems have been developed, in addition to task-specific methods developed for particular tasks, such as the morphological segmentation methods we have described so far. In this section, we will apply one such standard method, namely conditional random fields [Lafferty et al., 2001] to morphological segmentation, following Publication V and Publication VI. In Section 4.2 we reviewed how segmentation can be performed with a linear-chain conditional random field and discussed its parameter estimation and inference methods. As a generic method, condition random fields do not specify the exact details of how to perform segmentation, particularly what label set and feature set to apply. In this section, we will describe those in detail. Experimental results, however, will be deferred to Section 6.6, where we will compare all proposed methods.

A key difference between Morfessor and the CRF based approach is that Morfessor learns a *morph lexicon*, that is, it identifies the morphological units. In contrast, the key idea of the CRF approach is to focus the mod-

eling effort on **morph boundaries** instead of the whole segments. This is reflected in that the output variables of the CRF are labels that encode where the boundaries are, whereas for Morfessor the segmentation is encoded by sequence of morph strings.

We apply linear-chain conditional random fields to morphological segmentation by utilizing the averaged perceptron algorithm for parameter estimation [Collins, 2002] and Viterbi search to find the best segmentation for new words. The details of these procedures have been presented in Section 4.2. Next, we define the employed label and feature sets.

6.5.1 Label Set

The morphological segmentation task can be represented as a sequence labeling problem by assigning each character in a word to one of three classes, namely:

B	beginning of a multi-character morph
M	middle of a multi-character morph
S	single-character morph

Using this label set, one can represent the segmentation of the Finnish word *autoilta* (*from cars*) (*auto+i+lta*) as:

a	u	t	o	i	l	t	a
↓	↓	↓	↓	↓	↓	↓	↓
B	M	M	M	S	B	M	M

It can be noted that the minimal sufficient label set for segmentation would contain merely the labels B and M (see Section 4.2). The introduction of the label S enables modeling the particular properties of single character morphs. Preliminary experiments suggested that the presented label set provides an adequate trade-off for our learning setting which presupposes a small annotated data set.

6.5.2 Feature Set

The feature extraction function f captures the co-occurrence behavior of the label transitions (y_{t-1}, y_t) and a set of features describing character position t of word x. We employ binary indicator functions that describe the position t of word x using all left and right substrings up to a maximum length δ.

We first define *non-transition* features, which associate the left and right

substrings with label at position t, for example

$$f_i(y_{t-1}, y_t, \mathbf{x}, t) = \begin{cases} 1 & \text{if } y_t = l \text{ and} \\ & (x_t, x_{t+1}) = \text{'ed'} \\ 0 & \text{otherwise} \end{cases}, \tag{6.11}$$

for each label $l \in \{B, M, S\}$. The longest substring length δ is considered a hyper-parameter optimized on the development set. Second, we define *transition* features, which capture the transitional behavior between adjacent label positions:

$$f_j(y_{t-1}, y_t, \mathbf{x}, t) = \begin{cases} 1 & \text{if } (y_{t-1}, y_t) = (l, l') \\ 0 & \text{otherwise} \end{cases} \tag{6.12}$$

6.5.3 Leveraging Unannotated Data

In order to utilize unannotated data, we explore a straightforward approach based on feature set augmentation, introduced in Publication VI. We exploit *predictions of unsupervised segmentation algorithms* by defining *variants of the features* described in Section 6.5.2. The idea is to compensate the weaknesses of the CRF model trained on the small annotated data set using the strengths of the unsupervised methods that learn from large amounts of unannotated data.

For example, consider utilizing predictions of the unsupervised Morfessor algorithm [Creutz and Lagus, 2007] in the CRF model. In order to accomplish this, we first learn the Morfessor model from the unannotated training data, and then apply the learned model on the words in the annotated training set. Assuming the annotated training data includes the English word *drivers*, the Morfessor algorithm might, for instance, return a (partially correct) segmentation *driv* + *ers*. We present this segmentation to the CRF by defining a function $\mathcal{M}(t)$, which returns 0 or 1, if the position t is in the middle of a segment or in the beginning of a segment, respectively, as in

t	1	2	3	4	5	6	7
x_t	d	r	i	v	e	r	s
$\mathcal{M}(t)$	1	0	0	0	1	0	0

Now, given the function $\mathcal{M}(t)$, we define variants of the features in Expressions 6.11 and 6.12 by substituting the activation values of each feature function with the output of the function $\mathcal{M}(t)$. For example, we would

define a variant of the feature function f_i in (6.11) as

$$f_k(y_{t-1}, y_t, x, t) = \begin{cases} \mathcal{M}(t) & \text{if } y_t = l \text{ and} \\ & (x_t, x_{t+1}) = \text{'ed'} \\ 0 & \text{otherwise} \end{cases} \quad (6.13)$$

Using this approach, the CRF model learns to associate the output of the Morfessor algorithm in relation to the surrounding substring context. After defining the augmented feature set, the CRF model parameters can be estimated in a standard manner on the small, annotated training data set. Subsequent to CRF training, the Morfessor model is applied on the test instances in order to allow the feature set augmentation and standard decoding with the estimated CRF model. We expect the Morfessor features to specifically improve segmentation of compound words (for example, *brain+storm*), which are modeled with high accuracy by the unsupervised Morfessor algorithm [Creutz and Lagus, 2007], but are not present in the small annotated training set available for the supervised CRF training.

As another example of a means to augment the feature set, we make use of the fact that the output of the unsupervised algorithms, such as $\mathcal{M}(t)$, does not have to be binary (zeros and ones). To this end, we employ the classic letter successor variety (LSV) scores presented originally by Harris [1955].[3] The LSV scores utilize the insight that the predictability of successive letters should be high within morph segments, and low at the boundaries. Consequently, a high variety of letters following a prefix indicates a high probability of a boundary. We use a variant of the LSV values presented by Çöltekin [2010], in which we first normalize the scores by the average score at each position t, and subsequently logarithmize the normalized value. While LSV score tracks predictability given prefixes, the same idea can be utilized for suffixes, providing the letter predecessor variety (LPV). Subsequent to augmenting the feature set using the functions $LSV(t)$ and $LPV(t)$, the CRF model learns to associate high successor and predecessor values (low predictability) to high probability of a segment boundary. Appealingly, the Harris features can be obtained in a computationally inexpensive manner, as they merely require counting statistics from the unannotated data.

The feature set augmentation approach described above is computation-

[3] We also experimented on modifying the output of the Morfessor algorithm from binary to probabilistic, but these soft cues provided no consistent advantage over the standard binary output.

ally efficient if the computational overhead from the unsupervised methods is small. This is because the CRF parameter estimation is still based on the small amount of labeled examples. Meanwhile, the number of features incorporated in the CRF model (equal to the number of parameters) grows linearly in the number of exploited unsupervised algorithms. In other words, if the original number of CRF parameters is K, adding U unsupervised methods will result in a set of $(U+1)K$ parameters. Therefore, augmenting the CRF model using, for example, the $\mathcal{M}(t)$, $LSV(t)$, and $LPV(t)$ functions as described above would result in $4K$ parameters, which does not dramatically increase the cost of training.

6.6 Empirical Comparison of Semi-Supervised Methods for Morphological Segmentation

In this section we will summarize the work in semi-supervised learning of morphological analysis by comparing empirically the presented methods with other recent methods. The experiment includes data in four languages. Here, we will focus on morphological segmentation rather than analysis. As discussed in Section 2.2.2, direct evaluation of segmentation accuracy should be favored over the Morpho Challenge metric, as the latter has known weaknesses. Therefore, we will here employ the Boundary F1-score instead.

6.6.1 Experiments

In this section, we perform an empirical comparison of the Morfessor algorithms [Creutz and Lagus, 2002, 2005b, 2007], Publication IV, [Grönroos et al., 2014], the semi-supervised variant from adaptor grammar framework [Sirts and Goldwater, 2013], and the conditional random fields from Publication V and Publication VI all of which have freely available implementations for research purposes. The comparison was originally presented in Publication VII to extend the current literature on weakly supervised morphological segmentation by considering a wider range of methods and languages compared to previous work, and by providing an in-depth error analysis.

Data

We perform the experiments on four languages, namely, English, Estonian, Finnish, and Turkish. The English, Finnish, and Turkish data

	English	Estonian	Finnish	Turkish
train (unann.)	384,903	3,908,820	2,206,719	617,298
train (ann.)	1,000	1,000	1,000	1,000
devel.	694	800	835	763
test	10×1,000	10×1,000	10×1,000	10×1,000

Table 6.4. Number of word types in the data sets.

are from the Morpho Challenge 2009/2010 data set [Kurimo et al., 2009b, 2010]. The annotated Estonian data set is acquired from a manually annotated, morphologically disambiguated corpus.[4] Meanwhile, the unannotated words are gathered from the Estonian Reference Corpus [Kaalep et al., 2010]. Table 6.4 shows the total number of instances available for model estimation and testing.

Evaluation

For the overall evaluation we employ Boundary F1-score as described in Section 2.2.2. We employ type-based macro-averages and evaluate alternative analyses by choosing the best matching one.

Because we apply a different treatment of alternative analyses, the results reported in are not directly comparable to the boundary F1-scores reported for the Morpho Challenge competitions [Kurimo et al., 2009b, 2010]. A limited comparison can be made by noting that the best boundary F1-scores for all languages reported in Morpho Challenge 2007-2010 have been achieved with the semi-supervised Morfessor algorithm, and it is included in the current experiments.

Model Learning and Implementation Specifics

Here, we discuss the experimental setup and training regimes for the different algorithms. We defer the exact details of the training procedures to the presentation in Publication VII, but here we report on aspects relevant to the discussion.

Unsupervised Training Morfessor Baseline and Adaptor Grammars were trained in an unsupervised fashion following the data selection scheme employed by Sirts and Goldwater [2013]. Training is performed on data sets with only the most frequent words. The training set sizes are: 10k, 20k, 30k, 40k, 50k, 100k, 200k, 400k, ..., as well as the full set. The model with the best development set scores is then chosen. For AG the

[4] Available at `http://www.cl.ut.ee/korpused/morfkorpus/index.php?lang=en`

search is terminated at 50k. For Morfessor Baseline this scheme affects precision and recall similarly to the frequency cutoff discussed in Section 6.2.1. This training can be seen as unsupervised training with hyperparameter adjustment.

Semi-Supervised Morfessors We employ a recently released Python implementation of the Morfessor method [Virpioja et al., 2013, Smit et al., 2014].[5] The package implements both the unsupervised [Creutz and Lagus, 2002, 2007] and the semi-supervised Morfessor Baseline introduced in Publication IV. For Morfessor FlatCat we apply the Python implementation by Grönroos et al. [2014].[6] For both methods their hyperparameters α and β are optimized with grid search. Importantly, the Morfessor FlatCat segmentations are initialized with the output of the supervised CRF, as this was shown to be beneficial in experiments by Grönroos et al. [2014].

Adaptor Grammars For unsupervised AG learning we used the freely available implementation.[7] The metagrammar for AG Select is the same as in [Sirts and Goldwater, 2013]. Inductive learning with posterior grammar was done with a freely available CKY parser.[8] For unsupervised and semi-supervised AG, we used a three-level collocation-submorph grammar in which the final segmentation is parsed out as a sequence of Morphs:

$$\begin{aligned} \text{Word} &\rightarrow \underline{\text{Colloc}}^{+} \\ \underline{\text{Colloc}} &\rightarrow \underline{\text{Morph}}^{+} \\ \underline{\text{Morph}} &\rightarrow \underline{\text{SubMorph}}^{+} \\ \underline{\text{SubMorph}} &\rightarrow \text{Char}^{+} \end{aligned}$$

The AG models have a set of hyperparameters and these were all inferred automatically as described by Johnson and Goldwater [2009]. The semi-supervised AGs are also optimized for the amount of data as was described for the unsupervised training. The AG model is stochastic and each segmentation result is just a single sample from the posterior. As the preliminary experiments revealed a low variance between different samples, we report the results based on a single sample taken after running the sampler for 1000 Gibbs iterations and table label resampling turned on.

[5] Available at `https://github.com/aalto-speech/morfessor`
[6] Available at `https://github.com/aalto-speech/flatcat`
[7] Available at `http://web.science.mq.edu.au/~mjohnson/Software.htm`
[8] Also obtained from `http://web.science.mq.edu.au/~mjohnson/Software.htm`

Method	Train (ann.)	Train (unann.)	Pre.	Rec.	F1
English					
MORFESSOR BASELINE (PSV)	0	384,903	76.3	76.3	76.3
AG (PSV)	0	384,903	62.1	84.5	71.6
CRF (SV)	100	0	86.0	72.7	78.8
MORFESSOR BASELINE (SSV)	100	384,903	81.7	82.8	82.2
MORFESSOR FLATCAT (SSV)	100	384,903	83.6	83.0	83.3
AG (SSV)	100	384,903	66.0	87.0	75.0
AG SELECT (SSV)	100	384,903	75.9	79.4	77.6
CRF (SSV)	100	384,903	87.6	81.0	**84.2**
CRF (SV)	1,000	0	91.6	81.2	86.1
MORFESSOR BASELINE (SSV)	1,000	384,903	84.4	83.9	84.1
MORFESSOR FLATCAT (SSV)	1,000	384,903	86.9	85.2	86.0
AG (SSV)	1,000	384,903	72.1	85.0	78.0
AG SELECT (SSV)	1,000	384,903	76.7	82.3	79.4
CRF (SSV)	1,000	384,903	89.3	87.0	**88.1**

Table 6.5. Precision, recall, and F1-scores for the English data. The columns titled *Train (unann.)* denote the number of unannotated words utilized in learning. Meanwhile, the columns titled *Train (ann.)* denote the number of annotated words.

CRFs The employed Python implementation of the CRF model is based on the presentation of Publication V and Publication VI, as reviewed in Section 6.5.[9] For semi-supervised learning, we utilize log-normalized successor and predecessor variety scores and binary features extracted from the unsupervised Morfessor Baseline and AG described above.

6.6.2 Results

Segmentation accuracies for English, Estonian, Finnish, and Turkish are shown in Tables 6.5, 6.6, 6.7, and 6.8, respectively. First, we can contrast the supervised CRF with the unsupervised methods. Using merely 100 annotated instances, the supervised CRF achieves higher accuracies for English and Turkish compared to the unsupervised methods. With 1,000 annotated words the supervised CRF performs at a substantially higher segmentation accuracy compared to the unsupervised methods for all languages. This success of the supervised method is surprising given that the annotated training set contains a very small subset of the vocabulary.

When comparing the supervised CRF to the generative semi-supervised methods at 100 annotated words the semi-supervised Morfessors outper-

[9]Available at `http://users.ics.aalto.fi/tpruokol/`

Method	Train (ann.)	Train (unann.)	Pre.	Rec.	F1
Estonian					
MORFESSOR BASELINE (PSV)	0	3,908,820	76.4	70.4	73.3
AG (PSV)	0	3,908,820	59.3	86.4	70.3
CRF (SV)	100	0	79.2	59.1	67.7
MORFESSOR BASELINE (SSV)	100	3,908,820	77.0	76.1	76.5
MORFESSOR FLATCAT (SSV)	100	3,908,820	81.8	74.5	77.9
AG (SSV)	100	3,908,820	60.8	86.9	71.5
AG SELECT (SSV)	100	3,908,820	60.9	90.4	72.8
CRF (SSV)	100	3,908,820	81.5	82.1	**81.8**
CRF (SV)	1,000	0	88.4	76.7	82.1
MORFESSOR BASELINE (SSV)	1,000	3,908,820	80.6	80.7	80.7
MORFESSOR FLATCAT (SSV)	1,000	3,908,820	84.7	82.0	83.3
AG (SSV)	1,000	3,908,820	66.8	86.5	75.4
AG SELECT (SSV)	1,000	3,908,820	62.8	90.3	74.1
CRF (SSV)	1,000	3,908,820	90.2	86.3	**88.2**

Table 6.6. Precision, recall, and F1-scores for the Estonian data. The columns titled *Train (unann.)* denote the number of unannotated words utilized in learning. Meanwhile, the columns titled *Train (ann.)* denote the number of annotated words.

Method	Train (ann.)	Train (unann.)	Pre.	Rec.	F1
Finnish					
MORFESSOR BASELINE (PSV)	0	2,206,719	70.2	51.9	59.7
AG (PSV)	0	2,206,719	66.9	67.9	67.4
CRF (SV)	100	0	73.0	59.4	65.5
MORFESSOR BASELINE (SSV)	100	2,206,719	69.8	70.8	70.3
MORFESSOR FLATCAT (SSV)	100	2,206,719	77.6	73.6	75.5
AG (SSV)	100	2,206,719	69.2	69.3	69.3
AG SELECT (SSV)	100	2,206,719	66.8	73.6	70.0
CRF (SSV)	100	2,206,719	80.0	77.4	**78.7**
CRF (SV)	1,000	0	88.3	79.7	83.8
MORFESSOR BASELINE (SSV)	1,000	2,206,719	76.0	78.0	77.0
MORFESSOR FLATCAT (SSV)	1,000	2,206,719	81.6	80.2	80.9
AG (SSV)	1,000	2,206,719	73.4	73.6	73.5
AG SELECT (SSV)	1,000	2,206,719	69.4	74.3	71.8
CRF (SSV)	1,000	2,206,719	89.3	87.9	**88.6**

Table 6.7. Precision, recall, and F1-scores for the Finnish data. The columns titled *Train (unann.)* denote the number of unannotated words utilized in learning. Meanwhile, the columns titled *Train (ann.)* denote the number of annotated words.

Method	Train (ann.)	Train (unann.)	Pre.	Rec.	F1
Turkish					
MORFESSOR BASELINE (PSV)	0	617,298	67.9	65.8	66.8
AG (PSV)	0	617,298	72.0	76.0	74.0
CRF (SV)	100	0	84.6	71.8	77.7
MORFESSOR BASELINE (SSV)	100	617,298	76.6	80.5	78.5
MORFESSOR FLATCAT (SSV)	100	617,298	80.2	83.9	82.0
AG (SSV)	100	617,298	74.5	79.6	77.0
AG SELECT (SSV)	100	617,298	69.0	82.3	75.0
CRF (SSV)	100	617,298	81.3	86.0	**83.5**
CRF (SV)	1,000	0	90.0	87.3	88.6
MORFESSOR BASELINE (SSV)	1,000	617,298	85.1	89.4	87.2
MORFESSOR FLATCAT (SSV)	1,000	617,298	84.9	92.2	88.4
AG (SSV)	1,000	617,298	79.4	87.1	83.1
AG SELECT (SSV)	1,000	617,298	70.5	80.4	75.1
CRF (SSV)	1,000	617,298	89.3	92.0	**90.7**

Table 6.8. Precision, recall, and F1-scores for the Turkish data. The columns titled *Train (unann.)* denote the number of unannotated words utilized in learning. Meanwhile, the columns titled *Train (ann.)* denote the number of annotated words.

form the supervised CRF for all languages, and the AGs outperform the CRF for Estonian and Finnish. At 1,000 annotated words, the supervised CRF has surpassed all semi-supervised generative methods except for Morfessor FlatCat. Here, it should be noted that Morfessor FlatCat was initialized with the output of the supervised CRF. Therefore, the question is if the generative training from that starting point improves or reduces performance. For $1,000$ annotated instances, only the score for Estonian in improved.

Finally, for all tested languages and for both 100 and 1,000 annotated words, the semi-supervised CRF performs the best. For 1,000 annotated words, the margin to the second best is often quite large.

6.6.3 Error Analysis

The results given in the previous section raises questions that require some further analysis. First, the supervised CRF performs quite well despite operating based on a very limited part of the vocabulary. One can ask if the errors performed by the supervised CRF are different from those of the other methods that utilize the large unannotated training set. Second, since the performance of the semi-supervised CRF is far better than

its competition, what kind of errors does it manage to avoid?

To characterize the errors made by each method, we employ a categorization of morphs into the categories PREFIX, STEM, and SUFFIX, in addition defining a separate category for DASH. For the English and Finnish sections of the Morpho Challenge data set, the segmentation gold standard annotation contain additional information for each morph, such as part-of-speech for stems and morphological categories for affixes, that allow us to assign each morph into one of the morph type categories. In some rare cases the tagging is not specific enough, and we choose to assign the tag UNKNOWN. In particular, as we are evaluating segmentations, we lack the morph category information for the proposed analyses. Consequently, we cannot apply a straightforward category evaluation metric, such as category F1-score. In what follows, we instead show how to use the categorization on the gold standard side to characterize the segmentation errors.

We first observe that errors come in two kinds, **over-segmentation** and **under-segmentation**. In over-segmentation, boundaries are incorrectly assigned within morph segments, while in under-segmentation, the segmentation fails to uncover correct morph boundaries. For example, consider the English compound word *girlfriend* with a correct analysis *girl+friend*. Then, an under-segmentation error occurs in case the model fails to assign a boundary between the segments *girl* and *friend*. Meanwhile, over-segmentation errors take place if any boundaries are assigned within the two compound segments *girl* and *friend*, such as *g+irl* or *fri+end*.

As for the relationship between these two error types and the precision and recall measures in Equations (2.1) and (2.2), we note that over-segmentation solely affects precision, whereas under-segmentation only affects recall. This is evident as the measures can be written equivalently as:

$$\text{Precision} = \frac{C(\text{proposed}) - C(\text{over-segm.})}{C(\text{proposed})} = 1 - \frac{C(\text{over-segm.})}{C(\text{proposed})} \tag{6.14}$$

$$\text{Recall} = \frac{C(\text{reference}) - C(\text{under-segm.})}{C(\text{reference})} = 1 - \frac{C(\text{under-segm.})}{C(\text{reference})} \tag{6.15}$$

In the error analysis, we employ these equivalent expressions as they allow us to examine the effect of *reduction* in precision and recall caused by over-segmentation and under-segmentation, respectively.

The over-segmentation errors occur when a segment that should remain intact is split. Thus, these errors can be assigned into categories c according to the morph tags PREFIX, STEM, SUFFIX, and UNKNOWN. The

segments in the category DASH cannot be segmented and do, therefore, not contribute to over-segmentation errors. Under-segmentation errors occur when the proposed analysis lacks a boundary that exists in the gold standard segmentation. Their error categories are, therefore, given by the segment boundary categories in the gold standard segmentation, such as STEM-SUFFIX, STEM-STEM, and PREFIX-STEM. To simplify analysis, we have grouped all segment boundaries, in which either the left or right segment category is DASH into the CONTAINS DASH category. Boundary types that occur less than 100 times in the test data are merged into the OTHER category.

We decompose the precision and recall reductions in Equations (6.14) and (6.15) into those caused by errors in each category indexed by c and d:

$$\text{Precision} = 1 - \sum_{c} \frac{C(\text{over-segm.}(c))}{C(\text{proposed})} \tag{6.16}$$

$$\text{Recall} = 1 - \sum_{d} \frac{C(\text{under-segm.}(d))}{C(\text{reference})} \tag{6.17}$$

where c and d index the error categories for over-segmentation and under-segmentation, respectively.

Table 6.9 shows the occurrence frequency of each boundary category, averaged over alternative analyses. Evidently, we expect the total precision scores to be most influenced by over-segmentation of STEM and SUFFIX segment types because of their high frequencies. Similarly, the overall recall scores are expected to be most impacted by under-segmentation of STEM-SUFFIX and SUFFIX-SUFFIX boundaries. Finnish is also substantially influenced by the STEM-STEM boundary indicating that Finnish employs compounding frequently.

For simplicity, we employ a slightly different setup than is used for calculating the overall F1-score and, therefore, the numbers do not match exactly [10]

Next, we examine how different error types contribute to the obtained precision and recall measures, and consequently, the overall F1-scores. To this end, we discuss the error analyses for English and Finnish presented in Tables 6.10 and 6.11, respectively.

[10] When calculating the error analysis, we forgo the sampling procedure of taking 10×1000 words from the test set, employed for the overall F1-score for statistical significance testing, following Virpioja et al. [2011]. Rather, we calculate the error analysis on the union of these sampled sets.

Category	English		Finnish	
STEM	38608.8	(82.2%)	72666.0	(81.3%)
SUFFIX	7172.9	(15.3%)	15384.9	(17.2%)
PREFIX	1152.8	(2.5%)	946.5	(1.1%)
UNKNOWN	54.5	(0.1%)	414.0	(0.5%)
STEM-SUFFIX	5349.2	(62.6%)	9889.9	(45.8%)
SUFFIX-SUFFIX	1481.0	(17.3%)	5917.5	(27.4%)
STEM-STEM	613.4	(7.2%)	3538.0	(16.4%)
SUFFIX-STEM	n/a	n/a	1501.0	(6.9%)
CONTAINS DASH	458.0	(6.5%)	426.0	(2.0%)
PREFIX-STEM	554.3	(5.4%)	235.2	(1.1%)
OTHER	91.0	(1.1%)	105.4	(0.5%)

Table 6.9. Absolute and relative frequencies of the boundary categories in the error analysis. The numbers are averaged over the alternative analyses in the reference annotation.

Baselines The first two lines in Tables 6.10 and 6.11 present the baseline models WORDS and LETTERS. The WORDS model corresponds to an approach, in which no segmentation is performed, that is, all the words are kept intact. Meanwhile, the LETTERS approach assigns a segment boundary between all adjacent letters. These approaches maximize precision (WORDS) and recall (LETTERS) at the cost of the other. In other words, no model can produce more over-segmentation errors compared to LETTERS, whereas no model can produce more under-segmentation errors compared to WORDS.[11]

Morfessor Similarly to the baseline (WORDS and LETTERS) results, the majority of over-segmentation errors yielded by the Morfessor variants take place within the STEM and SUFFIX segments, while most under-segmentation errors occur at the STEM-SUFFIX and SUFFIX-SUFFIX boundaries. When shifting from unsupervised learning with MORF.BL (PSV) to semi-supervised learning with MORF.BL (SSV) and MORF.FC (SSV), the over-segmentation problems are alleviated rather substantially, resulting in higher overall precision scores. One also observes a dramatic increase in the overall recall scores indicating a smaller amount of under-segmentation taking place. However, the under-segmentation errors do

[11]Intuitively, WORDS should yield zero recall. However, when applying macro averaging, a word having a gold standard analysis with no boundaries yields a zero denominator and is therefore undefined. To correct for this, we interpret such words as having recall 1 which explains the non-zero recall for WORDS.

not decrease consistently: while the STEM-SUFFIX and SUFFIX-SUFFIX errors are decreased substantially, one additionally observes a decline or no change in the model's ability to uncover STEM-STEM and PREFIX-STEM boundaries.

Adaptor Grammars Similarly to the baselines and Morfessor results, the majority of over-segmentation errors yielded by the AG variants occur within the STEM and SUFFIX segments. Compared to the unsupervised AG (PSV), the first semi-supervised extension AG (SSV) manages to reduce over-segmentation of the STEM segments slightly and SUFFIX segments substantially, thus resulting in overall higher precision. Meanwhile, the second extension AG SELECT (SSV) also achieves overall higher precision by reducing over-segmentation of STEM segments substantially and SUFFIX segments slightly. Although both AG (SSV) and AG SELECT (SSV) improve recall on Finnish compared to AG (PSV), neither succeed in improving recall for English. Moreover, similarly to the Morfessor model family, one additionally observes a decline in the model's ability to capture STEM-STEM boundaries (AG (SSV)) and STEM-STEM and PREFIX-STEM boundaries (AG (SSV) and AG SELECT (SSV)).

Conditional Random Fields In contrast to the Morfessor and AG frameworks, the error patterns produced by the CRF approach do not directly follow the baseline approaches. Particularly, we note that supervised CRF (SV) approach successfully captures SUFFIX-SUFFIX boundaries and fails to find STEM-STEM boundaries, that is, behaves in opposite manner to the baseline results. CRF (SV) also under-segments the less frequent PREFIX-STEM and STEM-SUFFIX boundaries for English and Finnish, respectively. Meanwhile, the semi-supervised extension CRF (SSV) alleviates the problem of finding STEM-STEM boundaries substantially, resulting in improvement in overall recall. Note that improving recall means that CRF (SSV) is required to segment more compared to CRF (SV). For English, this increased segmentation results in a slight increase in over-segmentation of STEM, that is, the model trades off the increase in recall for precision.

6.6.4 Discussion

The most surprising result is the good performance of the supervised CRF. The error analysis showed that this good performance does not carry over to all classes of boundaries. This can be interpreted as the supervised training being very good at identifying frequent affixes and per-

	Over-Segmentation					Under-Segmentation						
Method	STEM	SUFFIX	PREFIX	UNKNOWN	**PRE / TOTAL**	STEM-SUFFIX	SUFFIX-SUFFIX	STEM-STEM	PREFIX-STEM	CONTAINS DASH	OTHER	**REC / TOTAL**
WORDS	0.0	0.0	0.0	0.0	100.0	55.1	8.6	5.9	4.4	2.5	0.6	23.1
LETTERS	71.1	11.8	1.7	0.3	15.1	0.0	0.0	0.0	0.0	0.0	0.0	100.0
MORF.BL (PSV)	20.6	2.9	0.0	0.1	76.4	17.0	4.7	0.6	1.1	0.0	0.2	76.4
MORF.BL (SSV)	14.3	1.3	0.1	0.0	84.4	9.8	0.6	2.8	2.1	0.0	0.4	84.3
MORF.FC (SSV)	11.2	1.7	0.0	0.1	87.1	8.6	0.5	2.2	2.5	0.1	0.4	85.5
AG (PSV)	31.9	5.6	0.1	0.1	62.3	10.7	3.5	0.1	0.7	0.2	0.1	84.7
AG (SSV)	26.2	1.4	0.1	0.1	72.3	11.5	0.9	0.7	1.2	0.3	0.3	85.0
AG SELECT (SSV)	18.4	4.8	0.0	0.1	76.6	8.2	1.4	2.2	4.1	1.5	0.4	82.2
CRF (SV)	7.3	0.9	0.1	0.0	91.8	10.4	0.5	4.2	2.9	0.1	0.4	81.5
CRF (SSV)	9.6	0.8	0.0	0.1	89.5	8.4	0.5	1.4	1.9	0.0	0.4	87.4

Table 6.10. Error analysis for English. Over-segmentation and under-segmentation errors reduce precision and recall, respectively. For example, the total precision of MORF. BL (PSV) is obtained as $100.0-20.6-2.9-0.0-0.1 = 76.4$. The lines MORF. BL (PSV), MORF. BL (SSV), and MORF. FC (SSV) correspond to the unsupervised Morfessor Baseline, semi-supervised Morfessor Baseline, and semi-supervised Morfessor FlatCat models, respectively.

forming worse on the less frequent morphs. For applications where compound splitting is required, this means that the unsupervised and semi-supervised generative models clearly have a place in addition to the supervised CRF.

The semi-supervised CRF, however, performs better than its competition for both suffixes and less frequent structure. Consequently, from a pragmatic point of view, that approach should be favored in the studied setting.

	Over-Segmentation					Under-Segmentation							
Method	STEM	SUFFIX	PREFIX	UNKNOWN	**PRE / TOTAL**	STEM-SUFFIX	SUFFIX-SUFFIX	STEM-STEM	SUFFIX-STEM	CONTAINS DASH	PREFIX-STEM	OTHER	**REC / TOTAL**
WORDS	0.0	0.0	0.0	0.0	100.0	49.2	21.8	17.2	4.8	1.4	1.0	0.6	4.1
LETTERS	65.2	13.8	0.7	0.6	19.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0
MORF.BL (PSV)	26.6	3.4	0.0	0.2	69.7	28.8	17.1	1.7	0.5	0.0	0.1	0.2	51.6
MORF.BL (SSV)	20.8	2.9	0.0	0.2	76.1	13.6	5.9	1.9	0.5	0.0	0.1	0.1	78.0
MORF.FC (SSV)	15.3	2.9	0.0	0.1	81.7	12.2	5.2	1.5	0.6	0.1	0.1	0.1	80.2
AG (PSV)	29.5	3.3	0.1	0.2	66.8	18.7	11.6	0.9	0.2	0.3	0.1	0.2	67.9
AG (SSV)	23.4	2.9	0.1	0.2	73.4	17.7	5.7	2.2	0.4	0.1	0.2	0.2	73.6
AG SELECT (SSV)	24.2	6.1	0.0	0.1	69.5	13.2	7.8	2.4	1.1	0.8	0.2	0.1	74.4
CRF (SV)	9.3	2.3	0.0	0.0	88.3	10.7	2.2	5.8	1.1	0.1	0.3	0.2	79.7
CRF (SSV)	9.2	1.4	0.0	0.1	89.3	8.0	2.3	1.2	0.4	0.1	0.1	0.2	87.8

Table 6.11. Error analysis for Finnish. Over-segmentation and under-segmentation errors reduce precision and recall, respectively. The lines MORF. BL (PSV), MORF. BL (SSV), and MORF. FC (SSV) correspond to the unsupervised Morfessor Baseline, the semi-supervised Morfessor Baseline, and semi-supervised Morfessor FlatCat models, respectively.

7. Conclusions

Learning morphology in an unsupervised fashion has been a popular problem in the past [Hammarström and Borin, 2011, Roark and Sproat, 2007, Creutz and Lagus, 2007, Goldsmith, 2001]. We have studied two novel approaches that improve over this work.

First, modeling allomorphy, that is, non-concatenative structure in morphology with string transformations. Second, utilizing weak supervision in the learning of morphology. The weak supervision was given in the form of annotated words and was provided in a semi-supervised setting in addition to an abundance of unannotated words.

From an algorithmic point of view we have presented extensions to the popular unsupervised morphological segmentation method Morfessor Baseline [Creutz and Lagus, 2002, 2007]. Our first extension, Allomorfessor, presented in Publication I and Publication II, extends the segmentation based model of Morfessor Baseline with string transformations, in order to model non-concatenative structure. We present experiments on four different languages and find empirically that the performance of the Allomorfessor method is very similar to the original Morfessor Baseline. We do, however, find a small but significant performance improvement for English and a small but significant performance reduction for Finnish, German, and Turkish. Moreover, we find that the model does not employ string transformations nearly to the extent suggested by the linguistic gold standard analysis, resulting in low recall. It appears, however, that the precision of the proposed string transformations is quite high, only rarely containing spurious analyses.

Regarding weak supervision we first developed efficient methods for hyperparameter adjustment for Morfessor Baseline in Publication III. Then in Publication IV we extended Morfessor Baseline to full semi-supervised learning. A contender for this generative modeling approach was devel-

oped in Publication V and Publication VI in the form of morphological segmentation with conditional random fields. We developed label sets and features sets, as well as semi-supervised training by augmenting the feature set with the output from unsupervised methods. In Publication VII we then compared the proposed methods to other recently proposed semi-supervised methods.

The empirical comparison shows that the semi-supervised CRF approach outperforms all other methods by a clear margin on all languages. Our goal was to improve the quality of morphological analysis over the unsupervised work while preserving the inexpensive nature of the original methods. In particular, we considered whether it is more cost effective to annotate data to provide weak supervision compared to further development of unsupervised methods. The empirical results show large improvements in accuracy over the unsupervised methods. When training with 1,000 annotated words, the improvements were 11.8, 14.9, 21.2, and 16.7 percentage points F1-score over the best unsupervised method for English, Estonian, Finnish, and Turkish, respectively. The method produces results with an F1-score in the range of 88-91% for all languages. Since annotating 1,000 words is not very time consuming, requiring at most some hours of effort, the results indicate that such weak supervision is very valuable. In contrast, improving a unsupervised model to a similar extent is likely to be quite demanding.

Furthermore, the error analysis performed provides insight into the relative strengths and weaknesses of the compared methods. Its results indicate that when the annotated data is employed only in a purely supervised way the overall accuracy can be quite high, but the accuracy for compound words and on prefixes is, nevertheless, worse than for methods that additionally utilize the unannotated data. We hypothesize that this is caused by stems and prefixes being open-class and, therefore, requiring large training data sets to be observed and, consequently also, learned. It appears, however, that the unannotated data can provide the required information to a large degree. For generative models, semi-supervised training appears to degrade performance for some error categories. In contrast, the semi-supervised CRF performs best overall without any underperforming error categories.

Bibliography

Malin Ahlberg, Markus Forsberg, and Måns Huldén. Semi-supervised learning of morphological paradigms and lexicons. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, 2014.

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. Polyglot: Distributed word representations for multilingual nlp. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning (CoNLL)*, 2013.

Ethem Alpaydin. *Introduction to machine learning*. MIT press, 2004.

Antti Arppe. The role of morphological features in distinguishing semantically similar words in finnish a study of cognitive verbs. In *Proceedings from the Corpus Linguistics Conference Series, Vol. 1, no. 1, Third Biennial Corpus Linguistics 2005 Conference*, Birmingham, UK, 2005.

R. Harald Baayen, Ton Dijkstra, and Robert Schreuder. Singulars and plurals in dutch: Evidence for a parallel dual-route model. *Journal of Memory and Language*, 37(1):94–117, 1997.

Marco Baroni, Johannes Matiasek, and Harald Trost. Unsupervised discovery of morphologically related words based on orthographic and semantic similarity. In *Proceedings of the ACL-02 workshop on Morphological and phonological learning*, pages 48–57, Morristown, NJ, USA, 2002. ACL.

Leonard E. Baum, Ted Petrie, George Soules, and Norman Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The annals of mathematical statistics*, pages 164–171, 1970.

Jean Berko. The child's learning of english morphology. *Word*, pages 150–177, 1958.

Delphine Bernhard. Simple morpheme labelling in unsupervised morpheme analysis. In *Advances in Multilingual and Multimodal Information Retrieval, 8th Workshop of the CLEF*, volume 5152 of *Lecture Notes in Computer Science*, pages 873–880. Springer Berlin / Heidelberg, 2008.

Delphine Bernhard. MorphoNet: Exploring the use of community structure for unsupervised morpheme analysis. In *Working notes for the CLEF 2009 Workshop*, Corfu, Greece, 2009.

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. ISBN 0387310738.

Bernd Bohnet, Joakim Nivre, Igor Boguslavsky, Richárd Farkas, Filip Ginter, and Jan Hajic. Joint morphological and syntactic analysis for richly inflected languages. *TACL*, 1:415–428, 2013.

Jan A. Botha and Phil Blunsom. Adaptor grammars for learning non-concatenative morphology. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 345–356, 2013.

Jan A. Botha and Phil Blunsom. Compositional Morphology for Word Representations and Language Modelling. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, Beijing, China, jun 2014.

Michael Brent. Minimal generative explanations: A middle ground between neurons and triggers. In *Proc. of the 15th Annual Meeting of the Cognitive Science Society*, pages 28–36, 1993.

Richard H. Byrd, Jorge Nocedal, and Robert B. Schnabel. Representations of quasi-newton matrices and their use in limited memory methods. *Mathematical Programming*, 63(1-3):129–156, 1994.

Victor Chahuneau, Eva Schlinger, Noah A. Smith, and Chris Dyer. Translating into morphologically rich languages with synthetic phrases. In *Proc. of EMNLP*, 2013.

Erwin Chan. *Structures and distributions in morphology learning*. PhD thesis, University of Pennsylvania, 2008.

Noam Chomsky and Morris Halle. *The sound pattern of English*. ERIC, 1968.

Michael Collins. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, volume 10, pages 1–8. Association for Computational Linguistics, 2002.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537, 2011.

Çagrı Çöltekin. Improving successor variety for morphological segmentation. In *Proceedings of the 20th Meeting of Computational Linguistics in the Netherlands*, 2010.

Mathias Creutz. Unsupervised segmentation of words using prior distributions of morph length and frequency. In *Proc. ACL'03*, pages 280–287, Sapporo, Japan, 2003.

Mathias Creutz and Krista Lagus. Unsupervised discovery of morphemes. In *Proceedings of the Workshop on Morphological and Phonological Learning of ACL'02*, pages 21–30, Philadelphia, Pennsylvania, USA, 2002.

Mathias Creutz and Krista Lagus. Induction of a simple morphology for highly-inflecting languages. In *Proc. 7th Meeting of the ACL Special Interest Group in Computational Phonology (SIGPHON)*, pages 43–51, Barcelona, July 2004.

Mathias Creutz and Krista Lagus. Inducing the morphological lexicon of a natural language from unannotated text. In *Proceedings of the AKRR'05*, Espoo, Finland, 2005a.

Mathias Creutz and Krista Lagus. Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0. Technical Report A81, Publications in Computer and Information Science, Helsinki University of Technology, 2005b.

Mathias Creutz and Krista Lagus. Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing*, 4(1), jan 2007.

Mathias Creutz, Teemu Hirsimäki, Mikko Kurimo, Antti Puurula, Janne Pylkkönen, Vesa Siivola, Matti Varjokallio, Ebru Arisoy, Murat Saraçlar, and Andreas Stolcke. Morph-based speech recognition and modeling of out-of-vocabulary words across languages. *ACM Transactions on Speech and Language Processing*, 5(1):3:1–3:29, December 2007.

Sajib Dasgupta and Vincent Ng. High-performance, language-independent morphological segmentation. In *The annual conference of the North American Chapter of the ACL (NAACL-HLT)*, 2007.

Hal Daumé III. Semi-supervised or semi-unsupervised? In *NAACL Workshop on Semi-supervised Learning for NLP*, 2009.

Adrià de Gispert, Sami Virpioja, Mikko Kurimo, and William Byrne. Minimum Bayes risk combination of translation hypotheses from alternative morphological decompositions. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 73–76, Boulder, USA, June 2009. Association for Computational Linguistics.

Carl G. de Marcken. *Unsupervised Language Acquisition*. PhD thesis, MIT, 1996.

Guy De Pauw and Peter Waiganjo Wagacha. Bootstrapping morphological analysis of gıkuyu using unsupervised maximum entropy learning. In *Proceedings of the Eighth Annual Conference of the International Speech Communication Association. Antwerp, Belgium*. Citeseer, 2007.

Hervé Déjean. Morphemes as necessary concept for structures discovery from untagged corpora. In *Proceedings of the Joint Conferences on New Methods in Language Processing and Computational Natural Language Learning*, pages 295–298. Association for Computational Linguistics, 1998.

Vera Demberg. A language-independent unsupervised model for morphological segmentation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL-07)*, June 2007.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B (Methodological)*, 39(1):1–38, 1977.

Markus Dreyer and Jason Eisner. Discovering morphological paradigms from plain text using a dirichlet process mixture model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 616–627. Association for Computational Linguistics, 2011.

Greg Durrett and John DeNero. Supervised learning of complete morphological paradigms. In *Proceedings of the 2013 Conference of the North American*

Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 1185–1195, Atlanta, Georgia, June 2013. Association for Computational Linguistics.

Chris Dyer. Using a maximum entropy model to build segmentation lattices for mt. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 406–414. Association for Computational Linguistics, 2009.

Steffen Eger. Sequence segmentation by enumeration: An exploration. *The Prague Bulletin of Mathematical Linguistics*, 100:113–131, 2013.

Yoav Freund and Robert E. Schapire. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296, 1999.

Dan Garrette and Jason Baldridge. Learning a part-of-speech tagger from two hours of annotation. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT-13)*, pages 138–147, Atlanta, GA, June 2013.

John Goldsmith. Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27(2):153–198, June 2001.

John Goldsmith. An algorithm for the unsupervised learning of morphology. *Natural Language Engineering*, 12(04):353–371, 2006.

Sharon Goldwater. *Nonparametric Bayesian Models of Lexical Acquisition*. PhD thesis, Brown University, 2006.

Sharon Goldwater and Mark Johnson. Priors in bayesian learning of phonological rules. In *Proceedings of the 7th Meeting of the ACL Special Interest Group in Computational Phonology: Current Themes in Computational Phonology and Morphology*, pages 35–42. Association for Computational Linguistics, 2004.

Sharon Goldwater and David McClosky. Improving statistical mt through morphological analysis. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 676–683. Association for Computational Linguistics, 2005.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. Interpolating between types and tokens by estimating power-law generators. In *Advances in Neural Information Processing Systems (NIPS)*, page 18, 2006.

Spence Green and John DeNero. A class-based agreement model for generating accurately inflected translations. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 146–155. Association for Computational Linguistics, 2012.

Stig-Arne Grönroos, Sami Virpioja, Peter Smit, and Mikko Kurimo. Morfessor FlatCat: An HMM-based method for unsupervised and semi-supervised learning of morphology. In *Coling 2014*, 2014.

Peter Grünwald. A tutorial introduction to the Minimum Description Length principle. In *Advances in Minimum Description Length: Theory and Applications*. MIT Press, 2005.

Nizar Habash and Fatiha Sadat. Arabic preprocessing schemes for statistical machine translation. In *Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, page 49, 2006.

Jan Hajič. Morphological tagging: Data vs. dictionaries. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 94–101. Association for Computational Linguistics, 2000.

Harald Hammarström and Lars Borin. Unsupervised learning of morphology. *Computational Linguistics*, 37(2):309–350, June 2011.

Zellig S. Harris. From phoneme to morpheme. *Language*, 31(2):190–222, 1955.

Martin Haspelmath. *Understanding morphology*. Arnold, London, 2002.

Martin Haspelmath and Andrea D. Sims. *Understanding morphology*. Hodder Education, 2nd edition edition, 2010.

Katri Haverinen, Jenna Nyblom, Timo Viljanen, Veronika Laippala, Samuel Kohonen, Anna Missilä, Stina Ojala, Tapio Salakoski, and Filip Ginter. Building the essential resources for finnish: the turku dependency treebank. *Language Resources and Evaluation*, 48(3):493–531, 2014.

Teemu Hirsimäki, Mathias Creutz, Vesa Siivola, Mikko Kurimo, Sami Virpioja, and Janne Pylkkönen. Unlimited vocabulary speech recognition with morph language models applied to Finnish. *Computer Speech and Language*, 20(4): 515–541, October 2006.

Charles F. Hockett. Two models of grammatical description. *Morphology: Critical Concepts in Linguistics*, 1:110–138, 1954.

Edwin T. Jaynes. *Probability Theory: The Logic of Science*. Cambridge University Press, 2003. ISBN 9780521592710.

Feng Jiao, Shaojun Wang, Chi-Hoon Lee, Russell Greiner, and Dale Schuurmans. Semi-supervised conditional random fields for improved sequence segmentation and labeling. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 209–216. Association for Computational Linguistics, 2006.

Mark Johnson and Sharon Goldwater. Improving nonparameteric Bayesian inference: experiments on unsupervised word segmentation with adaptor grammars. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 317–325. Association for Computational Linguistics, 2009.

Mark Johnson, Thomas L. Griffiths, and Sharon Goldwater. Adaptor grammars: a framework for specifying compositional nonparametric Bayesian models. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 641–648, Cambridge, MA, 2007. MIT Press.

Heiki-Jaan Kaalep, Kadri Muischnek, Kristel Uiboaed, and Kaarel Veskis. The Estonian reference corpus: Its composition and morphology-aware user interface. In *Proceedings of the 2010 Conference on Human Language Technologies*

– *The Baltic Perspective: Proceedings of the Fourth International Conference Baltic HLT 2010*, pages 143–146. IOS Press, 2010.

Ronald M. Kaplan and Martin Kay. Regular models of phonological rule systems. *Computational linguistics*, 20(3):331–378, 1994.

Fred Karlsson. *Yleinen Kielitiede*. Helsinki University Press, 2002.

Lauri Karttunen and Kenneth R. Beesley. Twenty-five years of finite-state morphology. *Inquiries Into Words, a Festschrift for Kimmo Koskenniemi on his 60th Birthday*, pages 71–83, 2005.

Samarth Keshava and Emily Pitler. A simpler, intuitive approach to morpheme induction. In *Proceedings of 2nd Pascal Challenges Workshop*, pages 31–35, 2006.

Özkan Kılıç and Cem Bozsahin. Semi-supervised morpheme segmentation without morphological analysis. In *Proceedings of the LREC 2012 Workshop on Language Resources and Technologies for Turkic Languages, Istanbul, Turkey*, 2012.

Alexandre Klementiev and Dan Roth. Weakly supervised named entity transliteration and discovery from multilingual comparable corpora. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 817–824. Association for Computational Linguistics, 2006.

Jan Kneissler and Dietrich Klakow. Speech recognition for huge vocabularies by using optimized sub-word units. In *INTERSPEECH*, pages 69–72, 2001.

Philipp Koehn and Kevin Knight. Empirical methods for compound splitting. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics-Volume 1*, pages 187–193. Association for Computational Linguistics, 2003.

Oskar Kohonen, Sami Virpioja, and Mikaela Klami. Allomorfessor: Towards unsupervised morpheme analysis. In *Evaluating Systems for Multilingual and Multimodal Information Access: 9th Workshop of the Cross-Language Evaluation Forum, CLEF 2008, Aarhus, Denmark, September 17-19, 2008, Revised Selected Papers*, volume 5706 of *Lecture Notes in Computer Science*, pages 975–982. Springer, 2009.

Oskar Kohonen, Sami Virpioja, Laura Leppänen, and Krista Lagus. Semi-supervised extensions to morfessor baseline. In *Proceedings of the Morpho Challenge 2010 Workshop*. Aalto University School of Science and Technology Faculty of Information and Natural Sciences Department of Information and Computer Science, Espoo, Finland, September 2010.

Andrei N. Kolmogorov. Three approaches to the quantitative definition ofinformation'. *Problems of information transmission*, 1(1):1–7, 1965.

Kimmo Koskenniemi. Two-level morphology: A general computational model for word-form recognition and production. Technical Report Publication 11, University of Helsinki, Department of General Linguistics, Helsinki, 1983.

Mikko Kurimo, Mathias Creutz, Matti Varjokallio, Ebru Arisoy, and Murat Saraclar. Unsupervised segmentation of words into morphemes – challenge 2005, an introduction and evaluation report. In *Proceedings of the 2nd Pascal Challenges Workshop*, Italy, 2006a.

Mikko Kurimo, Antti Puurula, Ebru Arisoy, Vesa Siivola, Teemu Hirsimäki, Janne Pylkkönen, Tanel Alumäe, and Murat Saraclar. Unlimited vocabulary speech recognition for agglutinative languages. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, HLT-NAACL '06, pages 487–494, Stroudsburg, PA, USA, 2006b. Association for Computational Linguistics.

Mikko Kurimo, Ville Turunen, and Matti Varjokallio. Overview of Morpho Challenge 2008. In *Evaluating Systems for Multilingual and Multimodal Information Access: 9th Workshop of the Cross-Language Evaluation Forum, CLEF 2008, Aarhus, Denmark, September 17-19, 2008, Revised Selected Papers*, volume 5706 of *Lecture Notes in Computer Science*, pages 951–966. Springer, 2009a.

Mikko Kurimo, Sami Virpioja, Ville Turunen, Graeme W. Blackwood, and William Byrne. Overview and results of Morpho Challenge 2009. In *Working Notes for the CLEF 2009 Workshop*, Corfu, Greece, September 2009b.

Mikko Kurimo, Sami Virpioja, Ville T. Turunen, Graeme W. Blackwood, and William Byrne. Overview and results of Morpho Challenge 2009. In *Working Notes for the CLEF 2009 Workshop*, Corfu, Greece, September 2009c.

Mikko Kurimo, Sami Virpioja, and Ville Turunen. Overview and results of Morpho Challenge 2010. In *Proceedings of the Morpho Challenge 2010 Workshop*, pages 7–24, Espoo, Finland, September 2010. Aalto University School of Science and Technology, Department of Information and Computer Science. Technical Report TKK-ICS-R37.

John Lafferty, Andrew McCallum, and Fernando C.N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289, 2001.

Yoong Keok Lee, Aria Haghighi, and Regina Barzilay. Modeling syntactic context improves morphological segmentation. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning (CoNLL)*, pages 1–9, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. ISBN 978-1-932432-92-3.

Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. In *Soviet physics doklady*, volume 10, page 707, 1966.

Constantine Lignos. Learning from unseen data. In Mikko Kurimo, Sami Virpioja, and Ville T. Turunen, editors, *Proceedings of the Morpho Challenge 2010 Workshop*, pages 35–38, Helsinki, Finland, September 2–3 2010. Aalto University School of Science and Technology.

Constantine Lignos, Erwin Chan, Mitchell P. Marcus, and Charles Yang. A rule-based acquisition model adapted for morphological analysis. In *Multilingual*

Information Access Evaluation Vol. I: 10th Workshop of the Cross-Language Evaluation Forum, CLEF 2009, Corfu, Greece, September 30 - October 2, 2009, Revised Selected Papers, Lecture Notes in Computer Science, pages 658–665. Springer, 2010.

Minh-Thang Luong, Richard Socher, and Christopher D. Manning. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning (CoNLL)*, pages 29–37. Association for Computational Linguistics, August 2013.

Gideon S. Mann and Andrew McCallum. Generalized expectation criteria for semi-supervised learning of conditional random fields. In *Proceedings of ACL-08: HLT*, pages 870–878. Association for Computational Linguistics, 2008.

Christoph D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, 1999.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

Peter H. Matthews. *Morphology*. Cambridge University Press„ 1991. ISBN 0-521-41043-6.

Robert McEliece. *The theory of information and coding*. Cambridge University Press, student edition edition, 2004.

Christian Monson, Jaime Carbonell, Alon Lavie, and Lori Levin. Paramor: Minimally supervised induction of paradigm structure and morphological analysis. In *Proceedings of Ninth Meeting of the ACL Special Interest Group in Computational Morphology and Phonology*, pages 117–125. Association for Computational Linguistics, 2007.

Christian Monson, Kristy Hollingshead, and Brian Roark. Simulating morphological analyzers with stochastic taggers for confidence estimation. In *Multilingual Information Access Evaluation I - Text Retrieval Experiments*, volume 6241 of *Lecture Notes in Computer Science*. Springer, 2010.

Thomas Müller, Helmut Schmid, and Hinrich Schütze. Efficient higher-order crfs for morphological tagging. In *In Proceedings of EMNLP*, 2013.

Jason Naradowsky and Sharon Goldwater. Improving morphology induction by learning spelling rules. In *IJCAI*, pages 1531–1536, 2009.

Karthik Narasimhan, Damianos Karakos, Richard Schwartz, Stavros Tsakalidis, and Regina Barzilay. Morphological segmentation for keyword spotting. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, 2014.

Gonzalo Navarro. A guided tour to approximate string matching. *ACM Computing Surveys*, 33(1):31–88, 2001.

Sylvain Neuvel and Sean A. Fulop. Unsupervised learning of morphology without morphemes. In *Proceedings of the ACL-02 workshop on Morphological and phonological learning-Volume 6*, pages 31–40. Association for Computational Linguistics, 2002.

Vincent Ng and Claire Cardie. Weakly supervised natural language learning without redundant views. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 94–101. Association for Computational Linguistics, 2003.

Marius Paşca. Weakly-supervised discovery of named entities using web search queries. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, pages 683–690, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-803-9.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

Tommi Pirinen. Automatic finite state morphological analysis of finnish language using open source resources (in finnish). Master's thesis, University of Helsinki, 2008.

Ari Pirkola. Morphological typology of languages for ir. *Journal of Documentation*, 57(3):330–348, 2001.

Hoifung Poon, Colin Cherry, and Kristina Toutanova. Unsupervised morphological segmentation with log-linear models. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 209–217. Association for Computational Linguistics, 2009.

Mirko Popovič and Peter Willett. The effectiveness of stemming for natural-language access to slovene textual data. *Journal of the American Society for Information Science*, 43(5):384–390, 1992.

Martin F. Porter. An algorithm for suffix stripping. *Program: electronic library and information systems*, 14(3):130–137, 1980.

Siyu Qiu, Qing Cui, Jiang Bian, Bin Gao, and Tie-Yan Liu. Co-learning of word representations and morpheme representations. In *Proceedings of the 25th International Conference on Computational Linguistics*, Dublin, Ireland, 2014.

Lawrence Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

Jason Riesa and David Yarowsky. Minimally supervised morphological segmentation with applications to machine translation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas (AMTA06)*, pages 185–192, 2006.

Jorma Rissanen. Stochastic complexity. *Journal of the Royal Statistical Society. Series B (Methodological)*, 49(3):223–239, 1987.

Jorma Rissanen. *Stochastic Complexity in Statistical Inquiry*, volume 15. World Scientific Series in Computer Science, Singapore, 1989.

Brian Roark and Richard William Sproat. *Computational approaches to morphology and syntax*. Oxford University Press Oxford, 2007.

Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

Edward Sapir. *Language, An Introduction To The Study Of Speech*. ERIC, 1921.

Patrick Schone and Daniel Jurafsky. Knowledge-free induction of morphology using latent semantic analysis. In *Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning*, pages 67–72, Morristown, NJ, USA, 2000. ACL.

Patrick Schone and Daniel Jurafsky. Knowledge-free induction of inflectional morphologies. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, pages 1–9. Association for Computational Linguistics, 2001.

Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 1948.

Miikka Silfverberg, Teemu Ruokolainen, Krister Lindén, and Mikko Kurimo. Part-of-speech tagging using conditional random fields: Exploiting sub-label dependencies for improved accuracy. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 259–264. Association for Computational Linguistics, 2014.

Kairit Sirts and Sharon Goldwater. Minimally-supervised morphological segmentation using adaptor grammars. *Transactions of the Association for Computational Linguistics*, 1(May):255–266, 2013.

Peter Smit, Sami Virpioja, Stig-Arne Grönroos, and Mikko Kurimo. Morfessor 2.0: Toolkit for statistical morphological segmentation. In *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 21–24, Gothenburg, Sweden, April 2014. Association for Computational Linguistics.

Benjamin Snyder and Regina Barzilay. Crosslingual propagation for morphological analysis. In *Proceedings of the AAAI*, pages 848–854, 2008a.

Benjamin Snyder and Regina Barzilay. Unsupervised multilingual learning for morphological segmentation. In *Proceedings of ACL-08: HLT*, pages 737–745, Columbus, Ohio, June 2008b.

Sebastian Spiegler and Peter A. Flach. Enhanced word decomposition by calibrating the decision threshold of probabilistic models and using a model ensemble. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 375–383, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

Sebastian Spiegler and Christian Monson. EMMA: A novel evaluation metric for morphological analysis. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING)*, August 2010.

Charles Sutton and Andrew McCallum. An introduction to conditional random fields for relational learning. *Introduction to statistical relational learning*, pages 93–128, 2006.

Charles Sutton and Andrew McCallum. An introduction to conditional random fields. *Foundations and Trends in Machine Learning*, 4(4):267–373, 2012.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):pp. 267–288, 1996.

Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining (IJDWM)*, 3 (3):1–13, 2007.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394. Association for Computational Linguistics, 2010.

Ville Turunen and Mikko Kurimo. Speech retrieval from unsegmented Finnish audio using statistical morpheme-like units for segmentation, recognition, and retrieval. *ACM Transactions on Speech and Language Processing*, 8(1):1:1–1:25, October 2011.

Sami Virpioja. *Learning Constructions of Natural Language: Statistical Models and Evaluations*. PhD thesis, Aalto University, December 2012.

Sami Virpioja and Oskar Kohonen. Unsupervised morpheme analysis with Allomorfessor. In *Working notes for the CLEF 2009 Workshop*, Corfu, Greece, 2009.

Sami Virpioja, Ville Turunen, Sebastian Spiegler, Oskar Kohonen, and Mikko Kurimo. Empirical comparison of evaluation methods for unsupervised learning of morphology. *Traitement Automatique des Langues*, 52(2):45–90, 2011.

Sami Virpioja, Peter Smit, Stig-Arne Grönroos, and Mikko Kurimo. Morfessor 2.0: Python implementation and extensions for Morfessor Baseline. Report 25/2013 in Aalto University publication series SCIENCE + TECHNOLOGY, Department of Signal Processing and Acoustics, Aalto University, Helsinki, Finland, 2013.

Andrew Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13 (2):260–269, September 1967.

Yang Wang, Gholamreza Haffari, Shaojun Wang, and Greg Mori. A rate distortion approach for semi-supervised conditional random fields. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2008–2016, 2009.

Richard Wicentowski and David Yarowsky. *Modeling and learning multilingual inflectional morphology in a minimally supervised framework*. PhD thesis, Johns Hopkins University, 2003.

David Yarowsky and Richard Wicentowski. Minimally supervised morphological analysis by multimodal alignment. In *Proceedings of the 38th Meeting of the ACL*, pages 207–216, 2000.

Shun-Zheng Yu. Hidden semi-markov models. *Artificial Intelligence*, 174(2):215–243, 2010.

Zhu Zhang. Weakly-supervised relation classification for information extraction. In *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management*, CIKM '04, pages 581–588, New York, NY, USA, 2004. ACM. ISBN 1-58113-874-1.

Xiaojin Zhu. Semi-supervised learning literature survey. *Computer Science, University of Wisconsin-Madison*, 2:3, 2006.

Xiaojin Zhu and Andrew B. Goldberg. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3(1):1–130, 2009.

George Kingsley Zipf. *Selective Studies and the Principle of Relative Frequency in Language*. Harvard University Press, Cambridge, MA, 1932.

Errata

Publication II

In Table 3, the C2 MAP numbers for Morfessor Baseline for English and Finnish are interchanged. The correct numbers are shown in the introductory part of this dissertation in Table 5.2. Moreover, the time complexity in Chapter 3.1 should be $O(K^2W)$ instead of $O(KW\log(W))$.

Publication IV

First, in Section 4 it was claimed that the semi-supervised extension could be easily applied to Morfessor Categories-MAP [Creutz and Lagus, 2005a, 2007]. This turned out not to be the case. For more discussion see Section 6.3.4. Second, it was claimed that there is no overlap between training and test set. Because of a programming error, discovered at the time of writing this, the sets were independently sampled with overlap. The biggest training set and the test set contain 16% overlap for English, and 1.6% for Finnish. This does, however, not invalidate the relative results, since the overlap is partial.

DISSERTATIONS IN INFORMATION AND COMPUTER SCIENCE

Aalto-DD171/2014 Suvitaival, Tommi
Bayesian Multi-Way Models for Data Translation in Computational Biology. 2014.

Aalto-DD177/2014 Laitinen, Tero
Extending SAT Solver with Parity Reasoning. 2014.

Aalto-DD178/2014 Gonçalves, Nicolau
Advances in Analysis and Exploration in Medical Imaging. 2014.

Aalto-DD191/2014 Kindermann, Roland
SMT-based Verification of Timed Systems and Software. 2014.

Aalto-DD207/2014 Chen, Xi
Real-time Action Recognition for RGB-D and Motion Capture Data. 2014.

Aalto-DD211/2014 Soleimany, Hadi
Studies in Lightweight Cryptography. 2014.

Aalto-DD28/2015 Su, Hongyu
Multilabel Classification through Structured Output Learning – Methods and Applications. 2015.

Aalto-DD31/2015 Talonen, Jaakko
Advances in Methods of Anomaly Detection and Visualization of Multivariate Data. 2015.

Aalto-DD43/2015 van Heeswijk, Mark
Advances in Extreme Learning Machines. 2015.

Aalto-DD62/2015 Luttinen, Jaakko
Bayesian Latent Gaussian Spatio-Temporal Models. 2015.

Morphological analysis decomposes complex words into smaller constituents. It is an important problem in natural language processing, particularly for morphologically rich languages whose large vocabularies make statistical modeling difficult. Morphological analysis has traditionally been approached with rule-based methods that are accurate, but expensive to produce. Unsupervised machine learning methods provide an inexpensive alternative, but their analyses are typically limited to concatenative morphology and less accurate than those of rule-based methods.
In this dissertation we study improvements to inexpensive methods for morphological analysis. We study extending the analysis of an unsupervised machine learning method to also include non-concatenative morphological phenomena. In addition, we examine if providing machine learning methods a small number of correctly analyzed examples improves accuracy enough to be cost-effective compared to developing better unsupervised models.

Aalto-DD 91/2015

ISBN 978-952-60-6270-9 (printed)
ISBN 978-952-60-6271-6 (pdf)
ISSN-L 1799-4934
ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)

Aalto University
School of Science
Department of Computer Science
www.aalto.fi

BUSINESS + ECONOMY

ART + DESIGN + ARCHITECTURE

SCIENCE + TECHNOLOGY

CROSSOVER

DOCTORAL DISSERTATIONS